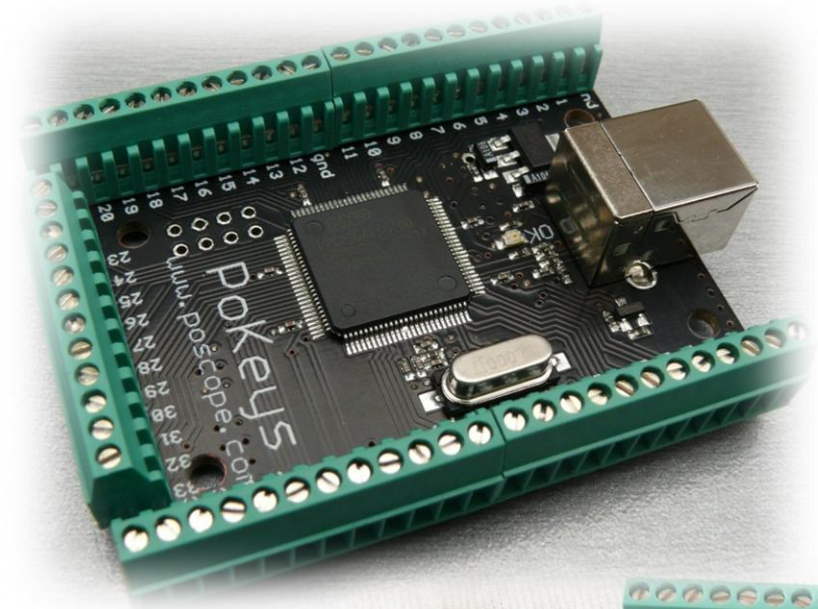
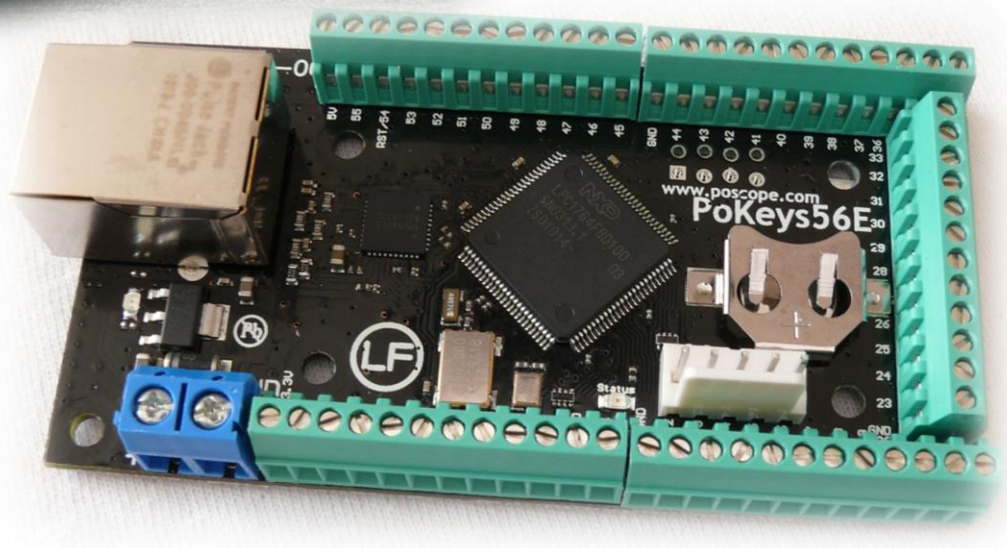
User's manual





PoKeys user's manual

1. Description....7
2. Features....8
PoKeys55....8
PoKeys56E....8
3. Requirements....9
3.1. USB devices....9
3.2. Network devices....9
4. Technical specifications....9
PoKeys55 dimensions....9
PoKeys56E dimensions....10
Electrical specification – limiting values....11
Electrical specification – static characteristic....11
Power supply for PoKeys56E....11
5. Installation....12
6. Using PoKeys configuration application....13
Inactive....13
Digital input....13
6.1.1. Direct key mapping – only for USB devices....13
Example:....13
6.1.2. Keyboard macro – only for USB devices....14
6.1.3. Encoder input....14
6.1.4. Matrix keyboard column selection....14
6.1.5. Digital counter (only on PoKeys56E)....14
Triggered digital input – only for USB devices....14
Digital output....15
6.1.6. Matrix keyboard row selection....15
6.1.7. Connection signal....15
Analog input....15
Analog output (PoKeys55 devices only)....15
Keyboard macros – only for USB devices....17
Displaying encoder RAW values....17
Enabling fast encoders....18

Joystick mapping – only for USB devices....18
Analog inputs and outputs dialog....20
Matrix keyboard settings....20
Using PWM module....23
HD44780-based character LCD....24
6.1.8. LCD settings....25
6.1.9. LCD operations....25
6.1.10. Display on/off settings....25
6.1.11. Custom characters....26
6.1.12. Move cursor....26
6.1.13. Print text....26
Matrix LED display settings....26
PoExtBus support....28
PoExtBus Pro (PoKeys56 devices only)....30
Adding new devices....30
PoEBkb....30
PoExtBus Pro/PoExtBus connection example....31
Changing User ID number....32
Saving current configuration to file....32
Example: starting a program on Windows using PoKeys55 device....32
7. PoKeys56E....34
Connecting to PoKeys56E device for the first time....34
Device discovery....34
Default settings....35
Connecting to device in other network....35
Security....35
Modbus support....36
Discrete inputs/outputs....36
Registers....37
Web interface....38
Connecting to PoKeys56E across the internet....38
8. Support for additional protocols....40
9. Communicating with the device using console (PoKeys55 devices only)....42
Supported operations....42

Enumerate PoKeys55 devices 42
Get details of the specific PoKeys55 device 42
Connect to PoKeys55 device 42
Save current configuration to flash memory 42
Get current pin setting 42
Set pin setting 43
Get digital input value 43
Set digital output value 43
Get analog input value 43
Set analog output value 43
10. Connecting common peripherals to PoKeys devices 45
11. Quick resetting the device configuration 51
12. Frequently asked questions 52
What software must be installed to operate the device? 52
I misconfigured the device. Now the device starts pressing virtual keys before I can do anything. What can I do? 52
How do I connect switch/relay/LED/… to PoKeys device? 52
I have two (or more) PoKeys devices connected on one system and cannot differentiate the devices to set the configurations. 52
It appears that pins 48 and 49 are floating. What should I do? 53
I have an encoder connected to the PoKeys device and when I read the raw data, the number climbs to 255 and then loops back around. Shouldn't the number continue to increase (or decrease depending on which way I turn the shaft)? 53
I have connected a switch to pin 4 and now PoKeys55 is not recognized by the computer anymore 53
I have connected a switch to pin 54 and now PoKeys55 is connecting in recovery mode on every boot. 53
There is spontaneous triggering of some of the pins. What is wrong? 54
My problem is that when I start the program, everything works good for about 3 to 5 seconds, then the CPU “lock up” and the only way to recover is to unplug the PoKeys device. 54
What can I connect to each pin of the PoKeys device? 55
13. PoKeys library functions 57
EnumerateDevices (for USB-based PoKeys devices) 57
ConnectToDevice (for USB-based PoKeys devices) 57
DisconnectDevice 57
StartEthernetDiscovery 58

StopEthernetDiscovery 58
GetNumberOfDetectedNetworkDevices 58
ConnectToNetworkDevice 58
GetDeviceID 59
GetDeviceIDEx 59
GetBuildDate 60
GetUserID 60
SetUserID 60
SetPinData 61
SetPinData 61
GetPinData 62
GetPinData 63
GetInput 63
SetOutput 64
GetAnalogInput 64
SetAnalogOutput 65
SaveConfiguration 65
GetMatrixConfiguration 66
SetMatrixConfiguration 67
GetPWMOutputs 68
SetPWMOutputs 69
LCDSetSettings 70
LCDInit 71
LCDClear 71
LCDGotoXY 71
LCDPutc 72
LCDPrint 72
LCDSetEntryMode 72
LCDDisplayOnOffControl 73
LCDDefineCustomCharacter 73
AuxilaryBusGetData 74
AuxilaryBusSetData 74
14. Interfacing with PoKeys library – C# example 75
Class initialization 75

Connecting to USB devices – PoKeys55 75
Connecting to Ethernet devices – PoKeys56E 76
Reading pin configuration 76
Reading pin key mapping (only on USB PoKeys devices – PoKeys55) 77
Reading input value 77
Joystick axis mapping (only on USB PoKeys devices – PoKeys55) 77
Block read - digital 78
Block read - analog 78
Block write - digital 78
Reading encoder RAW values 79
Macro operations (only on USB PoKeys devices – PoKeys55) 79
Display current time on LCD 80
Configuring PWM outputs 80
Configuring matrix keyboard 81
Reading matrix keyboard status 81
Write data to PoExtBus device 82
Write data to matrix LED display 82
Read tick counter 82
Read temperature from the LM75 sensor, connected to I^2C bus 82
Read temperature from DS18B20 sensor, connected to 1-wire bus 83
15. Major changes from 1.x to 1.7: 84
Pin 13 not functioning appropriately 84
Putting pin 4 low on startup disables PoKeys55 device from booting 84
16. Errata information 85
16.1.1. Pins 5 and 6 cannot be separately set as outputs/inputs 85
17. Grant of license 86

1. Description

PoKeys products line consists of simple, easy-to-use USB and network devices with the extended list of features making them powerful input/output devices. PoKeys55 devices also incorporate a virtual USB keyboard and joystick with a simple setup. PoKeys devices enable user to design specially built robust computer interfaces. The devices are highly adjustable and as such require no complex knowledge on device programming. Virtual USB keyboard supports emulation of single key presses or various programmable series of keys (keyboard macro sequences) while virtual USB joystick emulation supports mapping of analog inputs to joystick axes and mapping of digital inputs to joystick buttons.

If additional input and output capabilities are needed, the devices provide 55 digital 5V tolerant inputs or outputs, 5 10-bit analog inputs (on PoKeys55) / 7 12-bit analog inputs (on PoKeys56E) with adjustable software low-pass filter and one 10-bit analog output (on PoKeys55). These are complemented with 6 high-speed fully configurable PWM (pulse width modulation) outputs. User can freely set PWM period and PWM duty cycles. PWM module runs at 12 MHz (on PoKeys55) / 25 MHz (on PoKeys56E) and allows high-speed output switching.

They are controlled via included software, which enables user either to use the highly intuitive graphical user interface. Chosen settings can be stored on device, so no special software is needed on target system.

PoKeys devices also support up to 25 pairs of quadrature encoder signal inputs for which can be freely connected to any of the 55 inputs on PoKeys devices. These inputs increment or decrement the counters that can be read via provided software of other third-party applications. Changes in those signals can trigger virtual keyboard presses on USB versions of PoKeys devices. PoKeys55 device also supports three additional fast encoder inputs that provide support, when higher frequencies of the quadrature encoder signals are in use. PoKeys56E adds additional support for one ultra fast quadrature encoder signal that can handle even greater frequencies of these signals with ease.

In the cases when the application requires more than 55 inputs or outputs, PoKeys devices have the inbuilt support for matrix keyboards sized up to 16x8 with freely assignable key codes for virtual USB keyboard, while the PoExtBus bus feature adds the support for additional 80 digital inputs.

PoKeys devices also support hd44780-compatible character LCD displays and two serially driven 8x8 matrix LED displays.

Third-party application developers that are adding the support for PoKeys devices, are encouraged to use the supplied communication DLL that can be simply used in the different .NET framework based applications and various other programming languages that provide support for ActiveX interface.

To aid developers that are communicating with PoKeys devices on the low-level, the extensive documentation on device communication protocol can be downloaded free of charge from the product webpage.

2. Features

PoKeys55

- Compatible with USB 1.1/2.0 HID standard
- Standard USB keyboard simulation (with triggering support for up/down keys)
- Standard USB joystick simulation (6 axis, 32 buttons with triggering support)
- 55 digital inputs with pull-up resistors, freely mappable to virtual USB keyboard's keys
- 55 software controlled digital outputs
- 5 analog inputs (10-bit), freely mappable to any of virtual USB joystick axes (with adjustable low-pass filtering support)
- 1 software controlled 10-bit analog output, controlled via included software
- Up to 25 encoder pair inputs
- Up to 64 256-character long keyboard macro sequences
- Up to 16x8 matrix keyboard with triggered keys/alternate function support
- Two 8x8 matrix LED display support
- Up to 6 high-speed fully configurable PWM outputs support (12 MHz PWM timer)
- HD44780-based character LCD support (up to 4x20 characters)
- PoExtBus support for adding up to 10 external shift registers
- Support for Connection signal output
- Intuitive and user-friendly software
- Third-party support via communication DLL library and extensive protocol specification document that allows porting to other systems

PoKeys56E

- Ethernet 10/100 with DHCP client or fixed IP support
- TCP connection with device
- 55 digital inputs with pull-up resistors
- 55 software controlled digital outputs
- 7 analog inputs (12-bit) with adjustable low-pass filtering support
- Up to 26 encoder pair inputs (3 high-speed encoder inputs, 1 ultra high speed encoder input)
- Up to 16x8 matrix keyboard
- Two 8x8 matrix LED display support
- Up to 6 high-speed fully configurable PWM outputs support (25 MHz PWM timer)
- HD44780-based character LCD support (up to 4x20 characters)
- PoExtBus support for adding up to 10 external shift registers
- Modbus TCP support (access to digital IO, analog inputs, encoders' counters, digital counters values, PWM outputs, LCD display, LED matrix display, PoExtBus devices, matrix keyboard status)
- Support for communication with devices on I^2C and 1-wire buses
- Support for Connection signal output
- Communication compatible with PoKeys55 on application layer (data packet structure)
- Intuitive and user-friendly software
- Third-party support via communication DLL library and extensive protocol specification document that allows porting to other systems

3. Requirements

3.1. USB devices

- One available USB 1.1 or USB 2.0 port
- USB HID device driver enabled operating system (Windows 98 SE/ME/2000/XP/Vista, Linux, Mac OS)
- Included software requires Windows 2000/XP/Vista/Windows 7 with .NET framework 3.5 installed (ONLY FOR SYSTEMS WHERE THE DEVICES WILL BE CONFIGURED, TARGET SYSTEM NEEDS NO SOFTWARE INSTALLATION FOR THE DEVICE TO OPERATE AS A STANDARD USB KEYBOARD AND JOYSTICK).

3.2. Network devices

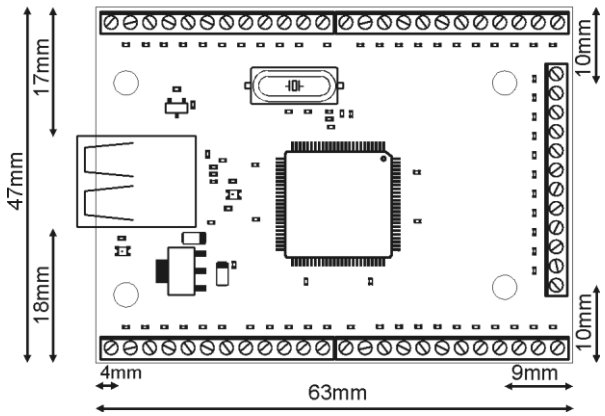
- Ethernet connection between host computer and PoKeys56E device
- 5V DC power supply with 400 mA. If any additional peripheral is connected to PoKeys56E, use appropriately more powerful power supply.
- Included software requires Windows 2000/XP/Vista/Windows 7 with .NET framework 3.5 installed (ONLY FOR SYSTEMS WHERE THE DEVICES WILL BE CONFIGURED, TARGET SYSTEM NEEDS NO SOFTWARE INSTALLATION FOR THE DEVICE TO OPERATE AS A STANDARD NEWTORK DEVICE).

4. Technical specifications

PoKeys55 dimensions

If that picture is printed with no scaling, it can be used as a marking tool for holes.

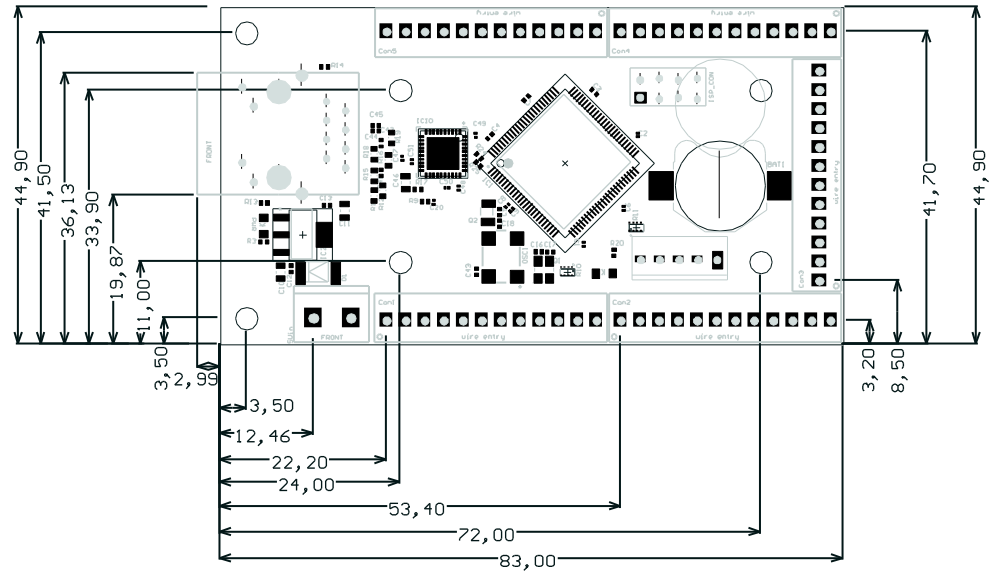
All holes are 3mm.

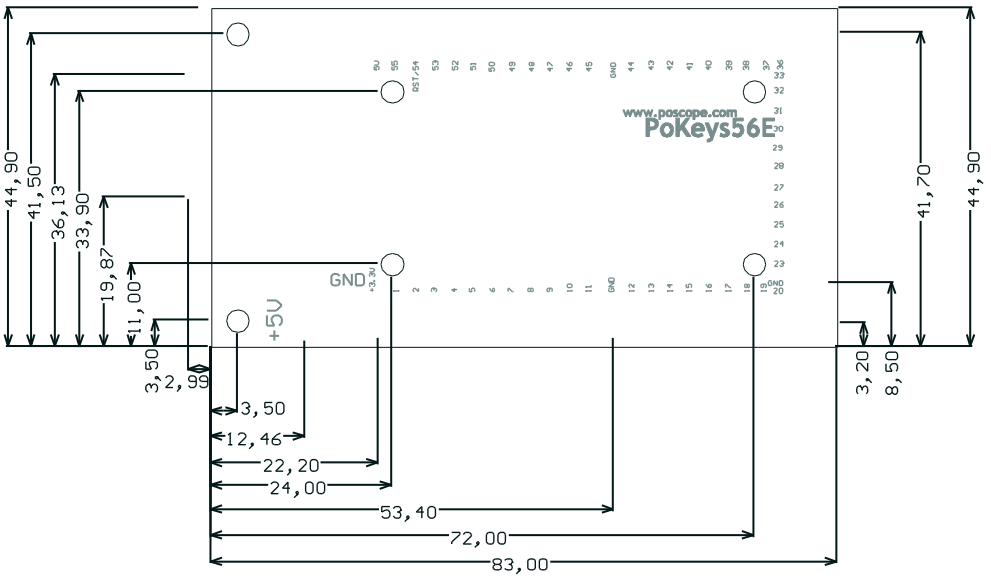


PoKeys56E dimensions

If that picture is printed with no scaling, it can be used as a marking tool for holes.

Measurements are in mm.





Electrical specification – limiting values

Symbol	Parameter	Min	Max	Unit
V_{IA}	analog input voltage on ADC related pins	-0.5	5.1	V
V_I	Input voltage on other pins	-0.5	6.0	V
V_{esd}	electrostatic discharge	-4000	4000	V

Electrical specification – static characteristic

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_I	input voltage	pin configured as digital input	0	-	5.5	V
V_O	output voltage	pin configured as digital output	0	-	3.3	V
V_{IH}	HIGH-level input voltage		2.0	-	-	V
V_{I_analog}	analog input voltage	pin configured as analog input	0	-	3.3	V
V_{O_analog}	analog output voltage	pin configured as analog output	0	-	3.3	V
V_{IL}	LOW-level input voltage		-	-	0.8	V
V_{hys}	hysteresis voltage		0.4	-	-	V
V_{OH}	HIGH-level output voltage	I_{OH} = -4 mA	2.9	-	-	V
V_{OL}	LOW-level output voltage	I_{OH} = 4 mA	-	-	0.4	V
I_{OH}	HIGH-level output current	V_{OH} = 2.9 V	-4	-	-	mA
I_{OL}	LOW-level output current	V_{OL} = 0.4 V	4	-	-	mA
I_{OHS}	HIGH-level short-circuit output current	V_{OH} = 0 V	-	-	-45	mA
I_{OLS}	LOW-level short-circuit output current	V_{OL} = 3.3V	-	-	50	mA
I_{pu}	pull-up current		0 (V_I >= 3.3V)	-50	-100	µA
I_{5V}	5V pin source current				400	mA
I_{5V}	5V power supply current for PoKeys56E **without peripherals**		250	300	400	mA

Power supply for PoKeys56E

PoKeys56E requires external 5V power supply to be connected to the board. If any additional peripheral is connected to PoKeys56E board, combined current of peripheral and PoKeys56E board must be taken into consideration when selecting the power supply.

Due to the protection diode in the circuit, a 0.8 V voltage drop can be observed between input voltage and 5V pin. When supplying the PoKeys56E board with 5V power supply, take care when connecting peripherals that require 5V supply voltage (e.g., LCD displays).

5. Installation

PoKeys55 is USB 1.1/2.0 compliant device and as such requires no additional drivers for operation as a standard USB keyboard and joystick.

PoKeys56E is an Ethernet device that requires an Ethernet connection between host computer and the device and 5V power supply. No additional drivers are required. Network firewalls must allow all traffic on IP/UDP port 20055.

To operate the device after the device has been configured there is no software installation necessary on a target system.

To configure the device the supplied software must be installed and the requirements listed in previous section of this manual must be met.

6. Using PoKeys configuration application

PoKeys settings application is the utility for setting up the device for normal use. Upon starting the program, the main window (Figure 1) with connection dialog is displayed. If there are PoKeys55 or PoKeys56E devices detected to be attached to the system, the device selection box will be populated with all devices. To easily identify a specific device, the User ID number is appended to the default device name. To start editing device settings, click the 'Connect' button. After the current configuration is uploaded from the device, the user interface is enabled (Figure 2).

There is graphical representation for configuration of each PoKeys device's pin on left and right side of main window. To change pin function, click on pin name and change its function in central 'Pin settings' frame.

There are 6 main pin functions possible: inactive, digital input, triggered digital input, digital output, analog input and analog output.

Inactive

Any pin can be set as inactive. Inactive pin is put in high-Z state with internal pull-up resistors enabled.

Digital input

Any one of the 55 pins can be configured as digital input by selecting 'Digital input' option box. All input pins have a weak pull-up resistor enabled and are 5V tollerant. If the pin polarity is inverted, check the 'Invert pin' box.

There are several additional possibilities for digital input pin functions.

6.1.1. Direct key mapping – only for USB devices

Digital input set up for direct key mapping acts like a keyboard key. When there is a high state on pin (on low state when using inverted option) PoKeys55 send a key associated with this pin. Select a keyboard key from drop-down box and check appropriate key modifiers (Shift, Ctrl, …).

Example:

Send Alt-F4: Select F4 from drop-down box and check Alt checkbox.

Send ((opening bracket): This key kombination differs from your system regional settings. As the PoKeys55 devices emulate a system keyboard, key associations depend on current sytem keyboard regional setting. To send an opening bracket symbol, one possible solution is to press Shift-8 (in most non-English countries) or to press Shift-9. Out of this reason there are no such secondary keys listed in drop-down box and must be entered by user as described above.

6.1.2. Keyboard macro – only for USB devices

If there is a need for more than one key to be sent on pin activation, there is a possibility to assign a keyboard macro to a pin. Please see section Keyboard macros for more information on editing and assigning keyboard macros.

6.1.3. Encoder input

Rotational encoder switch can be used with PoKeys device's digital inputs. It is possible to connect up to 25 encoders to one PoKeys device. To enable encoder input, first select encoder index with numerical up-down selector, then select appropriate encoder channel. The last step is to check the box 'Encoder'.

Same as simple digital inputs, encoders can be assigned to direct key mapping or keyboard macro (only with USB devices). This is possible for both directions (CW and CCW) separately. Simply set one mapping for channel A and another for channel B. To check for proper connection and settings, there is a special dialog that displays current encoder state. Please see section 'Displaying encoder RAW values'.

If needed, encoder inputs can be incremented or decremented 4x faster, therefore each complete step will produce increment or decrement of 4 sub-steps. Using this setting, higher precision can be obtained.

Encoder support should only be used for encoders that are hand-driven. It is not recommended to use encoders connected to driving axes of CNC milling machines.

6.1.4. Matrix keyboard column selection

Each digital input pin can be assigned as matrix keyboard column input. For additional matrix keyboard settings, see below.

6.1.5. Digital counter (only on PoKeys56E)

User can turn on digital counter option on selected inputs. On changes of the logical input signal, the pin digital counter is incremented. It is possible to select which changes affect the counter – rising or falling edges of the signal. Additional pin can be selected that selects whether counter is incremented or decremented when a rise or a fall of a signal is detected.

If switches are used in the combination with digital counters, external debouncing circuit must be installed.

Triggered digital input – only for USB devices

Any one of the 55 pins can be configured as triggered digital input by selecting 'Triggered input' option box. All input pins have a weak pull-up resistor enabled and are 5V tollerant. If the pin polarity is inverted, check the 'Invert pin' box.

This pin mode enabled user to select a key that is pressed only when a transition in a signal occurs. Different keys can be selected for 'LOW-to-HIGH' or 'Key up' event and for 'HIGH-to-LOW' or 'Key down' event. Selecting the keys is similar to Direct key mapping described above.

Digital output

Any one of the 55 pins can be configured as digital output by selecting ‘Digital output’ option box. Each pin can sink or source up to 4 mA of current, with the limitation that the pins combined source or sink current does not exceed 100 mA. If the polarity of the pin is inverted, check the 'Invert pin' box.

6.1.6. Matrix keyboard row selection

Each digital output pin can be assigned as matrix keyboard row output. For additional matrix keyboard settings, see below.

6.1.7. Connection signal

PoKeys55 devices can be set up to output a LOW or HIGH value on a pin when connection with the USB host is lost for more than 5 seconds. This option can be used to power down different peripheral devices when computer goes to sleep or is shut down (this function only works for systems that leave USB port powered after a shutdown).

Analog input

Analog input function is only available for pins 43 to 47 (on PoKeys55 devices) or for pins 41 to 47 (on PoKeys56E devices). On PoKeys55 devices, these analog inputs can also be freely mapped to any of the 6 joystick axis; X, Y, Z, rotation X, rotation Y and throttle. To monitor current analog input value please see section Analog input values box. To set up joystick mappings, go to Peripherals > Joystick settings.

To set low-pass filter for analog inputs, go to Peripherals > Analog inputs and outputs.

Analog output (PoKeys55 devices only)

Analog output function is only available for pin 43. It is possible to set analog output voltage for this pin with 10-bit resolution. To set the value of this pin, go to Peripherals > Analog inputs and outputs.

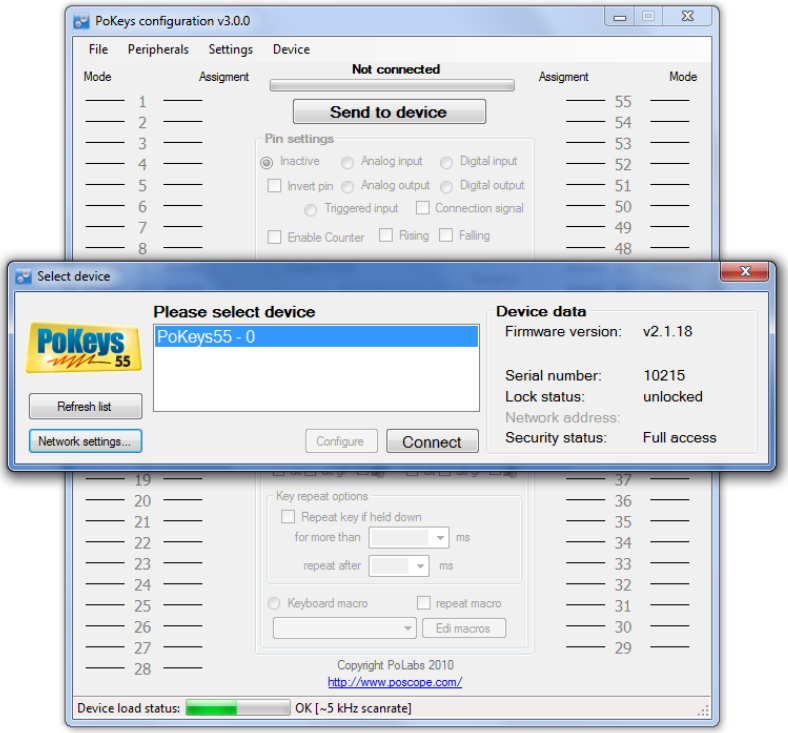


Figure 1: PoKeys configuration application startup

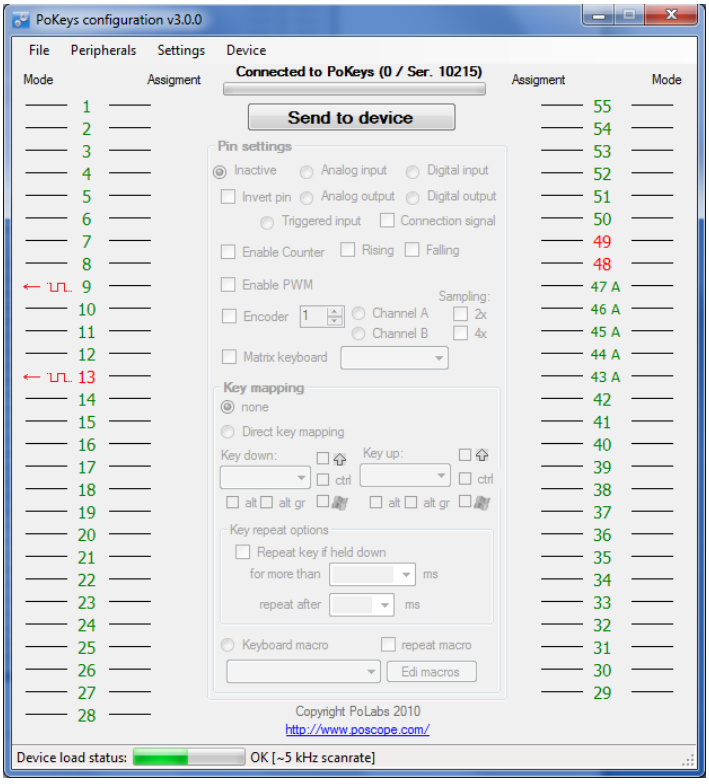


Figure 2: Main window

Keyboard macros – only for USB devices

PoKeys55 devices now support keyboard macros – the key press combinations that can be up to 256 keys long. To define a keyboard macro, first select Keyboard macro mapping option for one of the pins. 'Edit macros' and 'Get names' command buttons become enabled. To add, change or delete macros, click the 'Edit macros' button. The following dialog appears

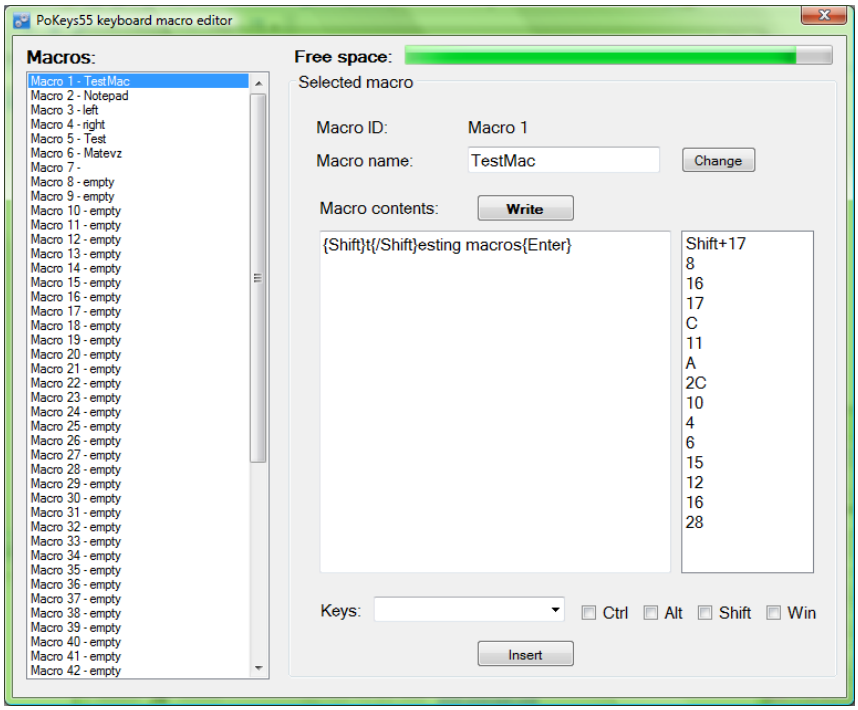


Figure 3: Macro editing dialog

First select the macro you want to edit. To change macro name, enter desired macro name (up to 7 characters long) in 'Macro name' text box and click 'Change' button. This name is used only to help user differentiate between multiple macros.

To set macro contents, simply enter text into 'Macro contents' text box. If there is an invalid character found, the text appears red. When finished, click Write to write macro to device.

List box at the right displays digital macro content.

Displaying encoder RAW values

To open encoder RAW values dialog, go to Peripherals menu and select 'Encoder RAW values'. The following dialog below appears. It simply shows the list of all encoders and their current values. In additional column, current encoder speed is displayed.

Status of the fast encoders is displayed in green, while the status of an ordinary encoder is displayed in light grey. Inactive encoders are displayed as dark grey boxes.

At the bottom of the window, there is a command button that can be used to reset the encoders' values.

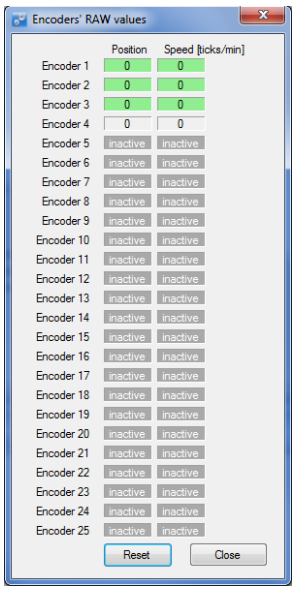


Figure 4: Encoders' RAW values

Enabling fast encoders

Sometimes finer control through encoders is desirable. To accomodate this need, PoKeys devices support 3 fast encoders inputs. To enable fast encoders, go to menu 'Peripherals' and select 'Enable fast encoders'. Fast encoders inputs are fixed to pins 1,2 for fast encoder input 1, pins 3,4 for fast encoder input 2 and pins 15,16 for fast encoder input 3.

Fast encoders' values are mapped directly to 'normal' encoders 1, 2 and 3. To read fast encoders RAW values, go to Peripherals menu and select 'Encoder RAW values', as described in previous chapter.

PoKeys56E devices support additional ultra fast encoder that provides support for even greater frequencies of quadrature encoder signals.

Joystick mapping – only for USB devices

Joystick axis and buttons mapping can be setup via Joystick mapping dialog. Got to 'Peripherals' and select 'Joystick settings…'. The dialog on Figure 5 appears.

Each axis can be assigned a analog input. In addition, analog to digital mapping option can be enabled. This allows user to connect an analog joystick to a PoKeys devices and simulate key presses for each direction of the joystick. To do so, first check 'Map to key' option. Then set the dead band (when input value will be between lower and upper dead band margins, no keys will be activated) using sliders. In the lower part of the window, select the mapping options.

For simple direct mapping (pin input status is directly reflected in joystick button status) use the 'Direct mapping' option and select pin number to be associated with selected joystick button. If more advanced behavior is needed (joystick button is pressed for a short time only on transitions of pin status), user should select 'Triggered mapping' option to select one pin that triggers selected joystick button on off-to-on transition (Down Event) and one pin that triggers this joystick button on on-to-off transition (Up Event).

If joystick button mapping is to be used in connection with encoder inputs, use 'Triggered mapping' option (encoder's values cannot be directly translated into direct mapping) and select a pin with the appropriate encoder channel. For example: pins 5 and 6 are set up as digital inputs with encoder (channel A on pin 5 and channel B on pin 6). When pin 5 is selected as Down event pin for joystick Button 2 and pin 6 is selected as Down event pin for joystick Button 3, rotating the encoder in positive direction will trigger joystick Button 2 on each detent. Similarly, rotating the encoder in negative direction, joystick Button 3 will be triggered on each detent. Up Event pin option cannot be used in connection with encoders.

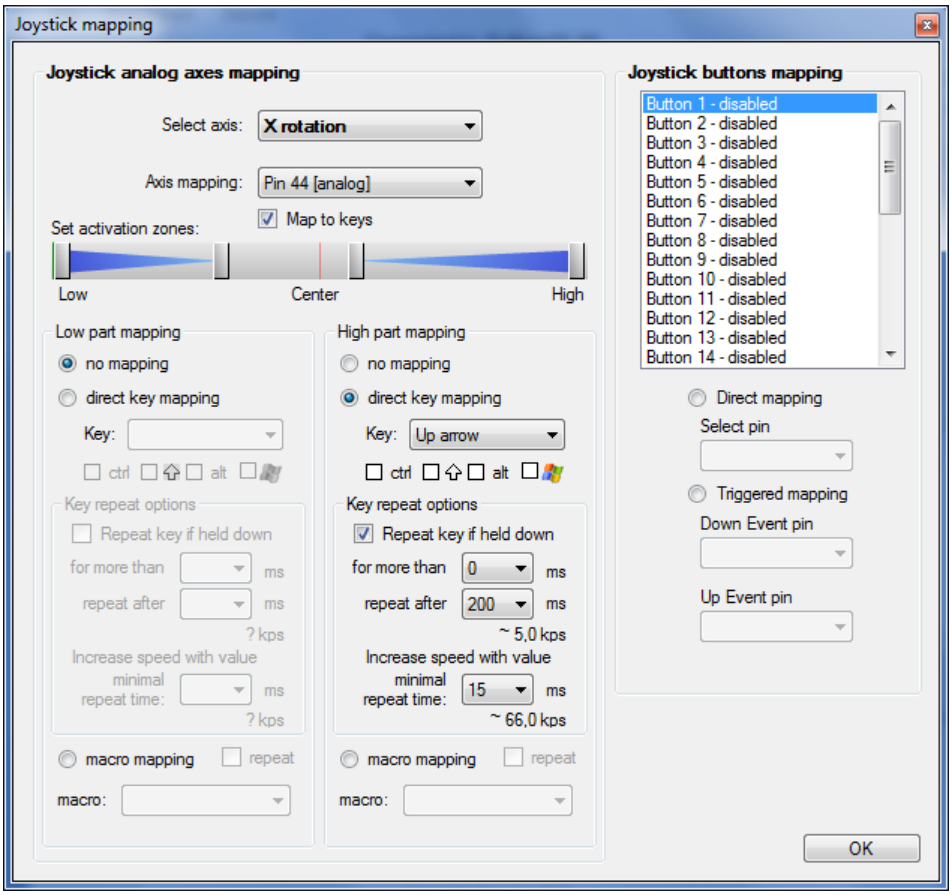


Figure 5: Joystick mapping settings

Analog inputs and outputs dialog

To open analog inputs dialog, go to Peripherals menu and select 'Analog inputs and outputs'. Dialog below appears. To enable display of analog input channel, check the appropriate check box. It is enabled only when the input is set up as analog input.

The progress bar displays the current voltage at the pin with the maximum at 3.3V. Below the input selection boxes user can set low-pass filtering for analog inputs. When analog input signal appears to be flickering or jumping due to analog signal noise, move the value for the filter to the right towards label 'slow signals' and then press Set button.

If pin 43 is set as analog output, analog value can be set for this pin.

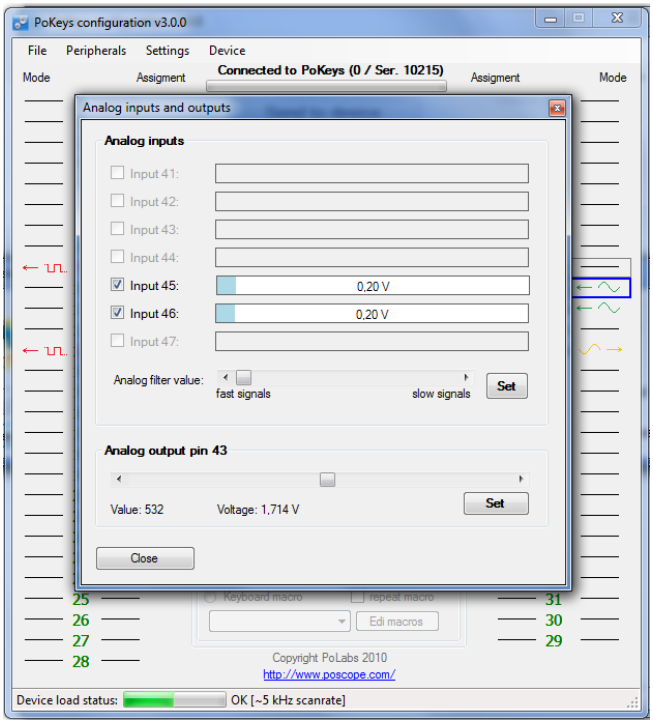


Figure 6: Analog inputs and outputs dialog

Matrix keyboard settings

To start using matrix keyboard, some insight is needed into working of a matrix keyboard. Matrix keyboard is a set of buttons, connected in a mesh. All buttons in a row share one contact, same goes for each of the buttons in the column. If a button is pressed, a key press is detected with a periodic scanning of each of the rows and columns. PoKeys devices use digital outputs for setting the voltage levels on rows and read column voltage levels using digital inputs that already have internal pull-up resistors, so no external circuitry is needed.

PoKeys devices support matrix keyboards of up to 16x8 in size, simpler 3x3, 4x3, 4x4 and others are of course fully supported.

Now, let us look how to set up a 4x3 keyboard. Open Peripherals > Matrix keyboard and set number of rows to 4 and number of columns to 3. Now click 'Enable matrix keyboard'. Matrix keyboard is schematically drawn below. Even lower, key mapping settings can be selected (on PoKeys55 devices). Associate keys as it is needed in your application and close window. Now, select 3 pins and set them as digital inputs (column pins) and 4 pins as digital outputs (row pins). When finished, click Send to device button and start using your matrix keyboard.

To test the matrix keyboard, set the matrix keyboard settings, assigns proper pins and send the configuration to the device. Then open the matrix keyboard settings window again and press keys on the matrix keyboard. Detected keys are displayed in the graphical representation of the matrix keyboard with a colored background.

On PoKeys55 devices, to set different key presses for 'key press' and 'key release' events, check 'Triggered mapping' option and select different settings for 'Down key' ('key press' event) and 'Up key' ('key release' event).

Figure 7: Standard 4x3 matrix keyboard

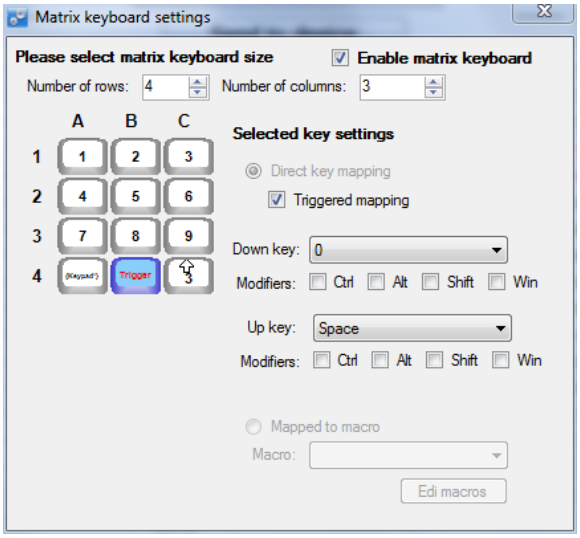


Figure 8: Matrix keyboard configuration

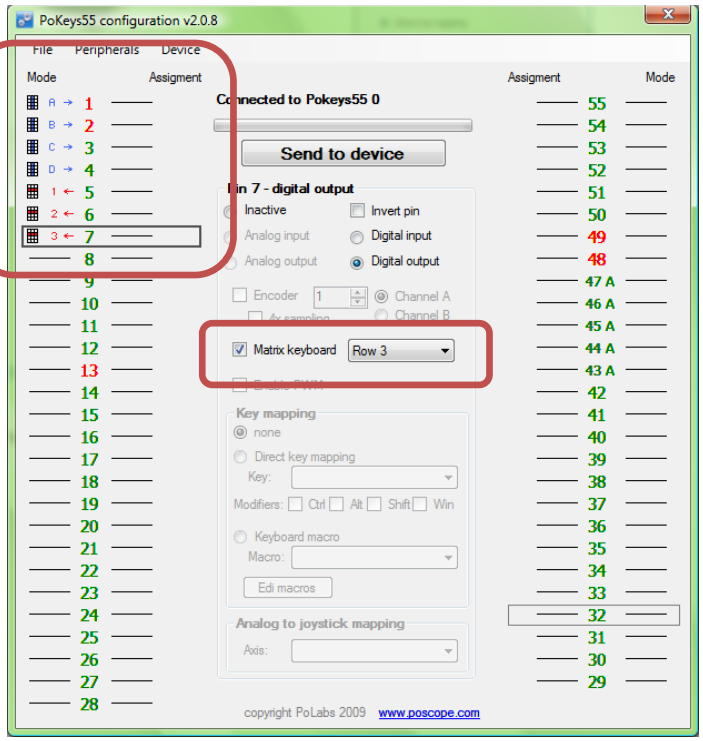


Figure 9: Assigning row and column pins

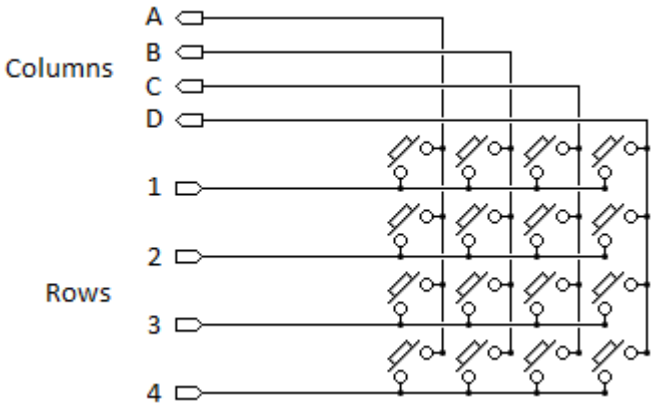


Figure 10: Matrix keyboard internal structure

PoKeys devices support matrix keyboard with alternate function keys. User can assign two different keyboard keys to each of the matrix keyboard buttons. If Fn+ key input pin is inactive, then the default function key is simulated. If the Fn+ key input pin is activated, an alternate function key is simulated instead of default.

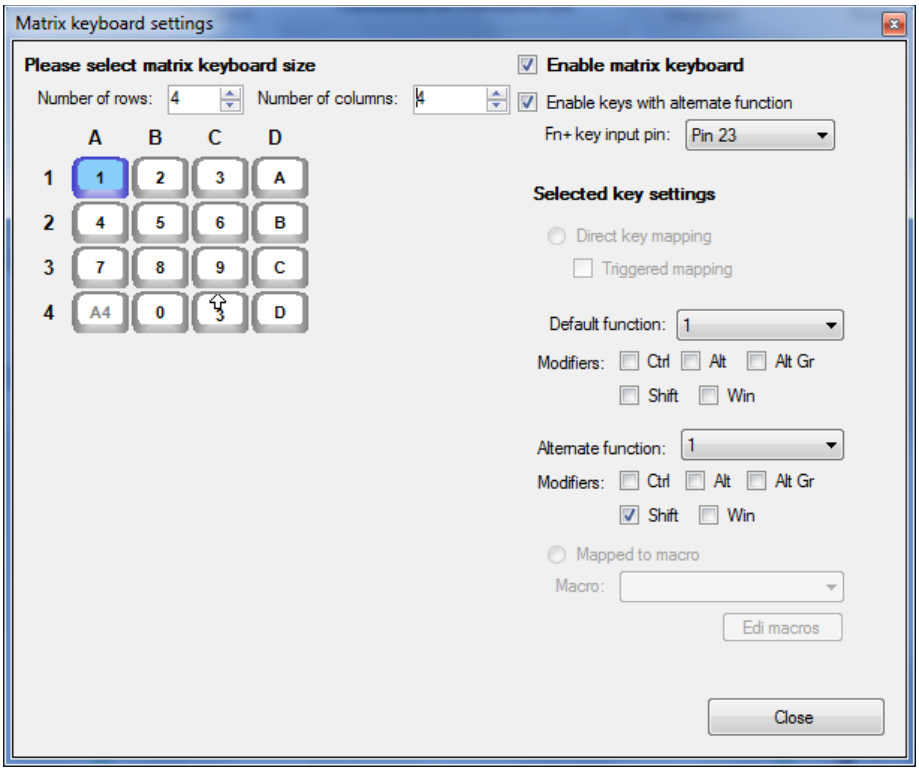


Figure 11: Matrix keyboard with alternate function enabled

Using PWM module

PoKeys device's PWM (pulse width modulation) module can be set up via Peripherals > PWM outputs….

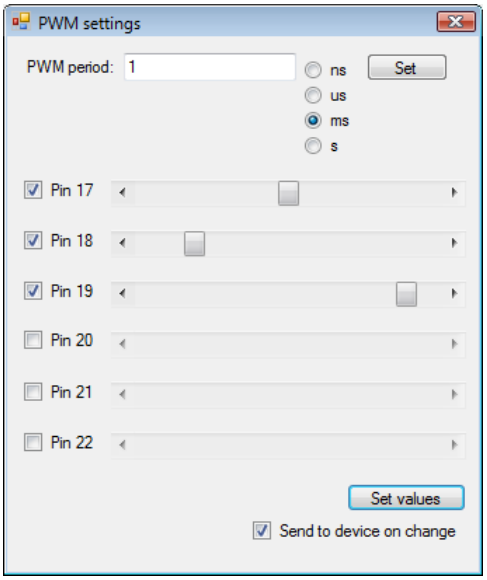


Figure 12: PWM outputs settings

In this window, user can enter PWM period and set PWM duties for each channel. Channels can be independantly enabled or disabled. After a change is made, user must click 'Set values' button or

check 'Send to device on change' checkbox. Left position of a slider means 0% and right position 100% respectively.

HD44780-based character LCD

User can connect almost any widely available character LCD that is based on HD44780 or similar chipset.

Figure 13: Typical 2x16 character LCD

Usually these displays come in various sizes - 1/2/4 line with 8/16/20 characters and colors (black letters on green background, white letters on blue background ...).

These displays share standard pin-out that is listed in the table below:

Pin	Symbol	Function	PoKeys pin
1	**Vss**	Ground	**GND**
2	**Vdd**	Positive supply (usually 5V)[1]	**5V (usually) or 3.3V**
3	**Vo**	Contrast adjustment	**Variable resistor between GND and supply or PWM output**
4	**RS**	Instruction/data input	**Pin 29**
5	**R/W**	Read/write	**Pin 28**
6	**E**	Enable signal	**Pin 30**
7	DB0	Data bus – bit 0	Not connected
8	DB1	Data bus – bit 1	Not connected
9	DB2	Data bus – bit 2	Not connected
10	DB3	Data bus – bit 3	Not connected
11	**DB4**	Data bus – bit 4	**Pin 26 (or secondary 34)**
12	**DB5**	Data bus – bit 5	**Pin 25 (or secondary 33)**
13	**DB6**	Data bus – bit 6	**Pin 24 (or secondary 32)**
14	**DB7**	Data bus – bit 7	**Pin 23 (or secondary 31)**
15	Backlight (optional)		
16	Backlight (optional)		

Table 1: LCD pin assignments

[1] Positive supply voltage depends on LCD used. User should find this information in datasheet of the LCD in use. **On PoKeys56E take special care that the power supply voltage for the LCD is adequate (there is 0.8 V of voltage drop between power supply input and 5V pin).**

LCD Display can be used to display various data. A third-party application or a script can execute all supported operations, including LCD initializing, clearing, moving cursor, setting display shifting mode, custom character defining and of course showing text. All this is available through easy-to-use PoKeysDevice DLL interface.

Functions of this interface can be tested through PoKeys settings application. Just open Peripherals > Test LCD… and dialog below will appear.

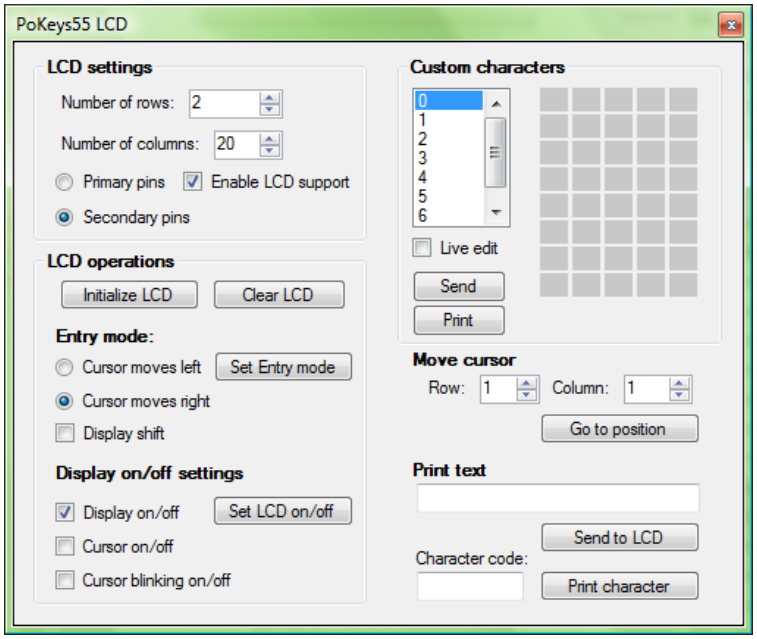


Figure 14: Character LCD testing dialog

6.1.8. LCD settings

In this part, user can set number of rows and columns in the LCD used. Support for LCD can be enabled or disabled also. Data pins for LCD can be selected on primary (23 to 26) or secondary (31 to 34) pins. ***Secondary pins must be selected in case of matrix LED display 2 in use.***

6.1.9. LCD operations

Before user can start using the LCD, LCD module must be initialized. This is done via 'Initialize LCD' button. Button 'Clear LCD' clears LCD display and moves cursor to home position.

User can also set entry mode settings of LCD module. Cursor can be set-up to move either right (normally) or left after each character displayed. If 'Display' shift is enabled, whole display shifts with every new character displayed.

Settings are processed after user clicks button 'Set Entry mode'.

6.1.10. Display on/off settings

User can set on/off switches for whole display, cursor and cursor blinking.

Settings are processed after user clicks button 'Set LCD on/off'.

6.1.11. Custom characters

Simple interface enables to draw up to 8 custom characters. These characters can then be used on display. Selecting 'Live edit' mode will transfer the character each time a change is made to any of the pixels. Character can be previewed via button 'Print', which puts current custom character on the LCD display.

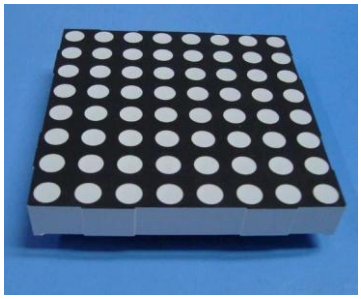
6.1.12. Move cursor

This section enabled user to move cursor to any position on the screen.

6.1.13. Print text

Sends entered text to display module. If advanced characters are needed, enter character code in lower text box and press 'Print character'.

Matrix LED display settings



User can connect one or two matrix LED displays on PoKeys device. Additional circuitry is needed to deserialize data, used to refresh matrix LED display.

Once enabled, PoKeys device assigns pins 9, 10 and 11 to be used with display 1, and pins 23, 24, 25 to be used with display 2.

Function	Matrix LED display 1	Matrix LED display 2
data	9	23
latch	10	24
clock	11	25

Table 2: Matrix LED displays pin assignments

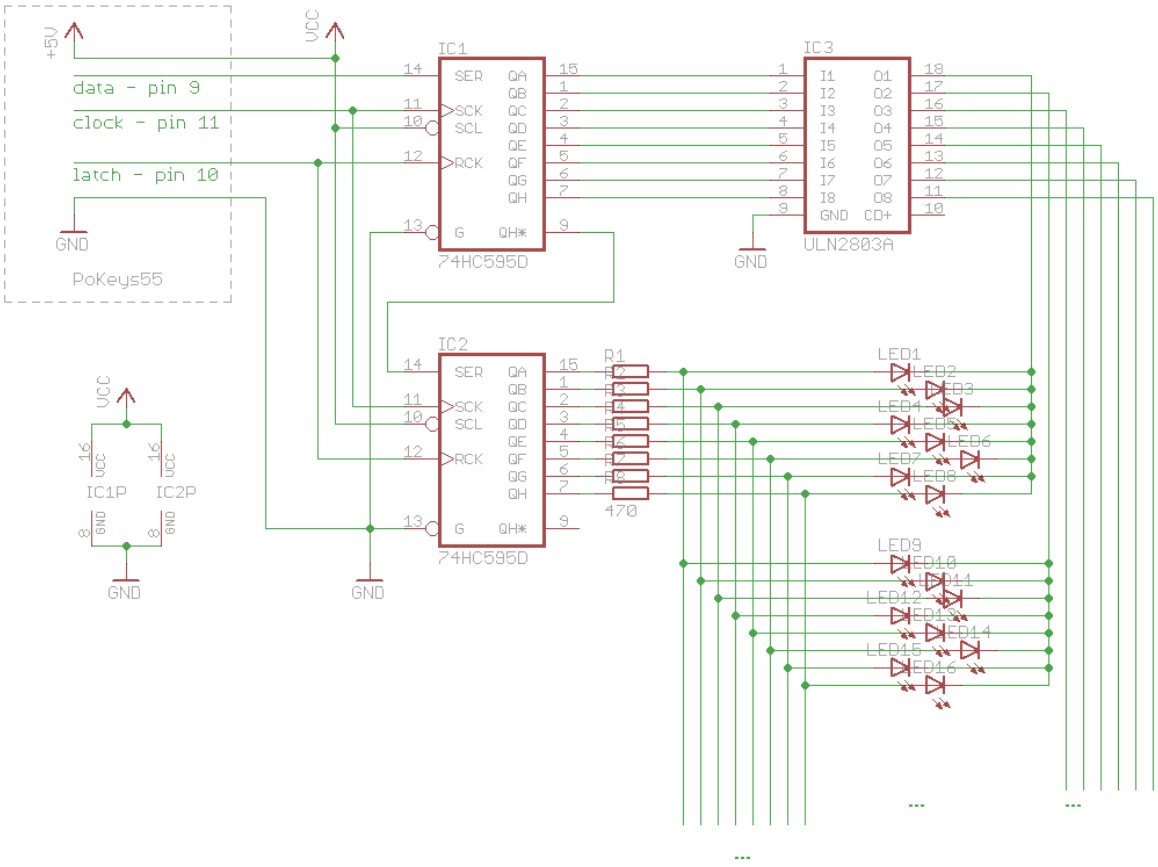


Figure 15: Matrix LED display deserialization circuit

To set-up matrix LED displays, open 'Peripherals' > 'LED displays'. The following dialog appears

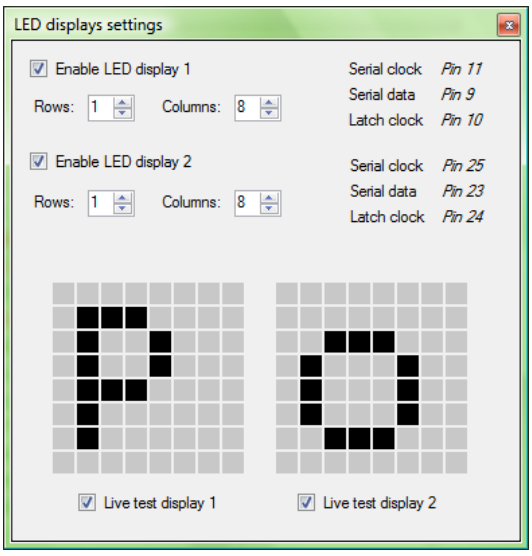


Figure 16: Matrix LED setup dialog

For each of the displays, number of rows and columns can be selected. On the bottom, after enabling 'Live test display x', user can test the displays in real-time by clicking on gray rectangles. Left-click turn-s selected pixel on, while right-click resets selected pixel.

PoExtBus support

PoExtBus bus support enables user to add additional 10 8-bit shift registers to the project based on PoKeys device. This gives additional 80 digital outputs that can be easily controlled with included dll interface or other 3rd party interface for PoKeys device.

On PoKeys55, device assigns pins 35, 36 and 37 to be used with PoExtBus. On PoKeys56 series, there is a dedicated connector on the board, which serves for the PoExtBus functions.

Function	ExtBus – PoKeys pin
Clock	35
Data	36
Latch	37

Table 3: PoExtBus pin assignments

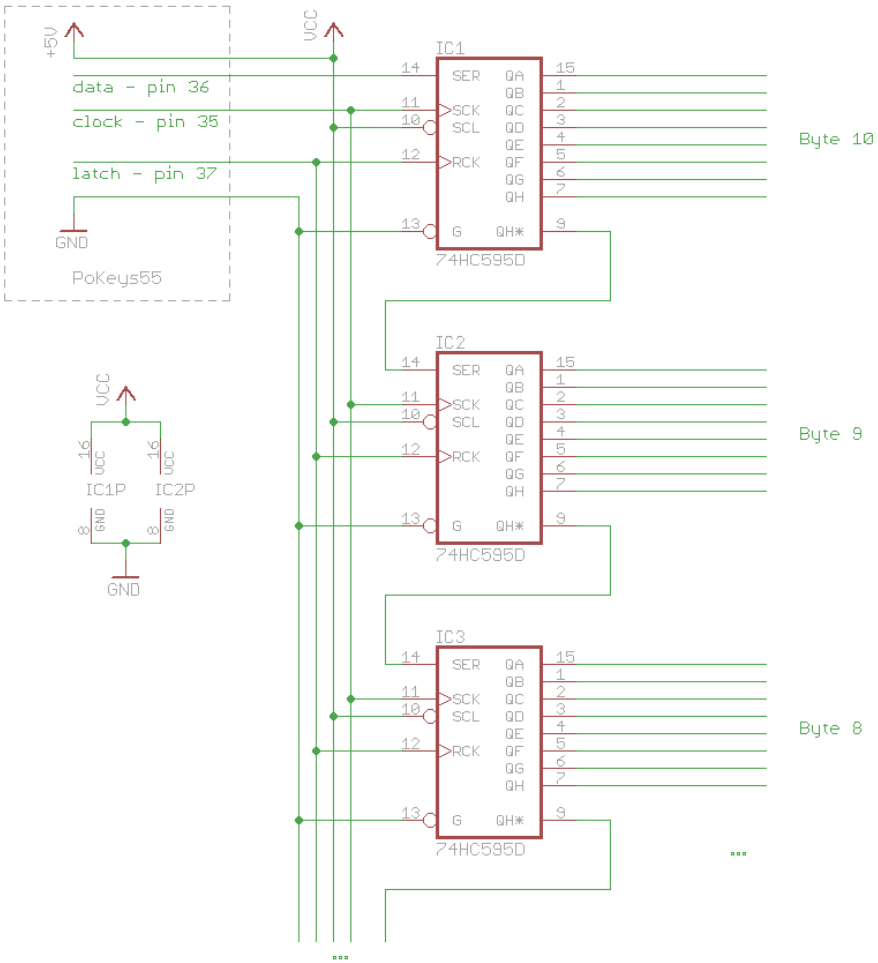


Figure 17: PoExtBus deserialization circuit

To set-up and test PoExtBus, open 'Peripherals' > PoExtBus...'. The dialog below (Figure 18) appears. By using mouse left and right clicks, user can turn on or off each of the outputs.

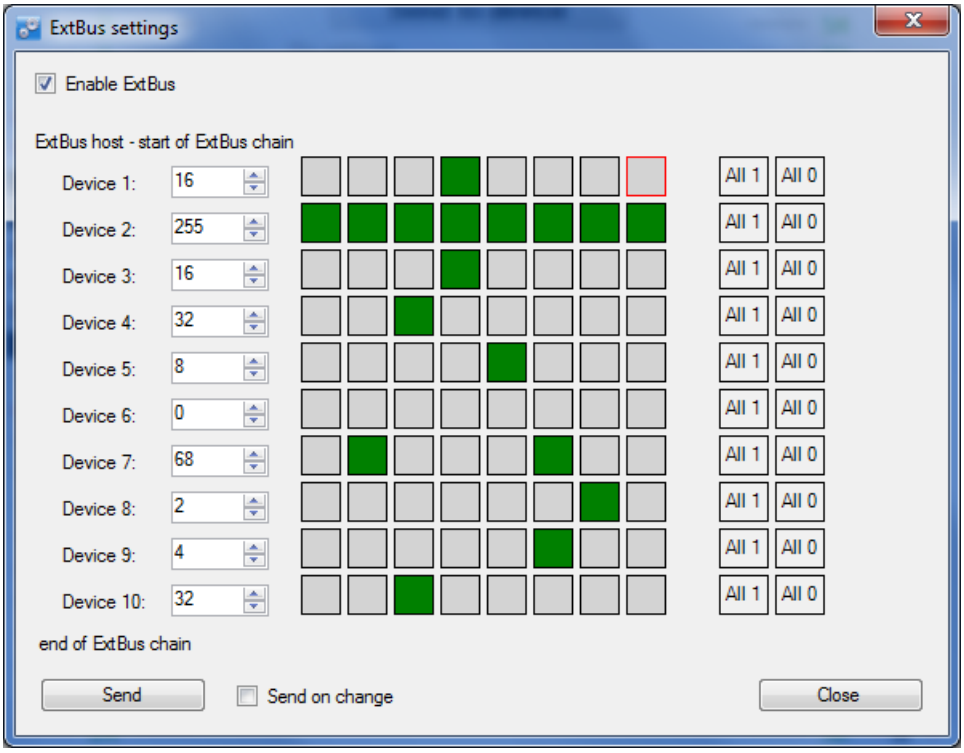


Figure 18: PoExtBus setup dialog

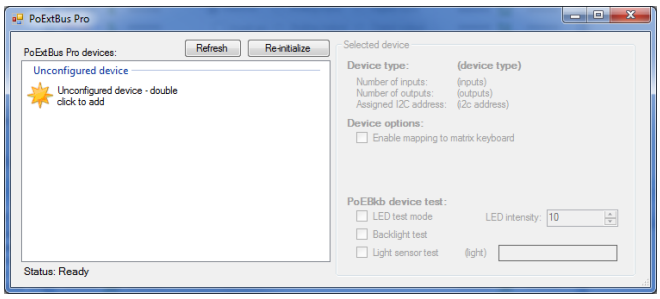
PoExtBus Pro (PoKeys56 devices only)

PoKeys56 devices support PoExtBus Pro devices that extend PoKeys56 functionality. Multiple PoExtBus Pro and PoExtBus devices can be linked together. While the PoExtBus Pro devices and PoExtBus devices share the same ExtBus connector, found on PoKeys56 boards, user should pay attention in connecting devices of both types together (see the schematics below).

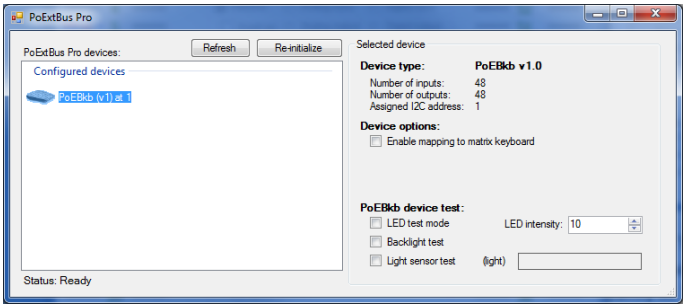
All PoExtBus Pro devices must be connected in parallel to each other and directly to the PoKeys56 board, while the PoExtBus devices should be connected in series after the PoExtBus Pro devices.

Adding new devices

After connecting new PoExtBus Pro device, go to Peripherals>PoExtBus Pro.... The following dialog will appear

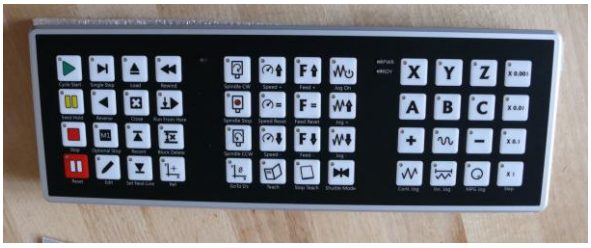


In order to register new device, double click on the 'Unconfigured device' icon. In the next 10 seconds press any key on the device that is about to be added. If the process is successful, status LED on the device will stop blinking and will be constantly lit. The device will also be listed as in the dialog below:



PoEBkb

PoEBkb devices are variously configured keyboards with optional signal LEDs under each key.



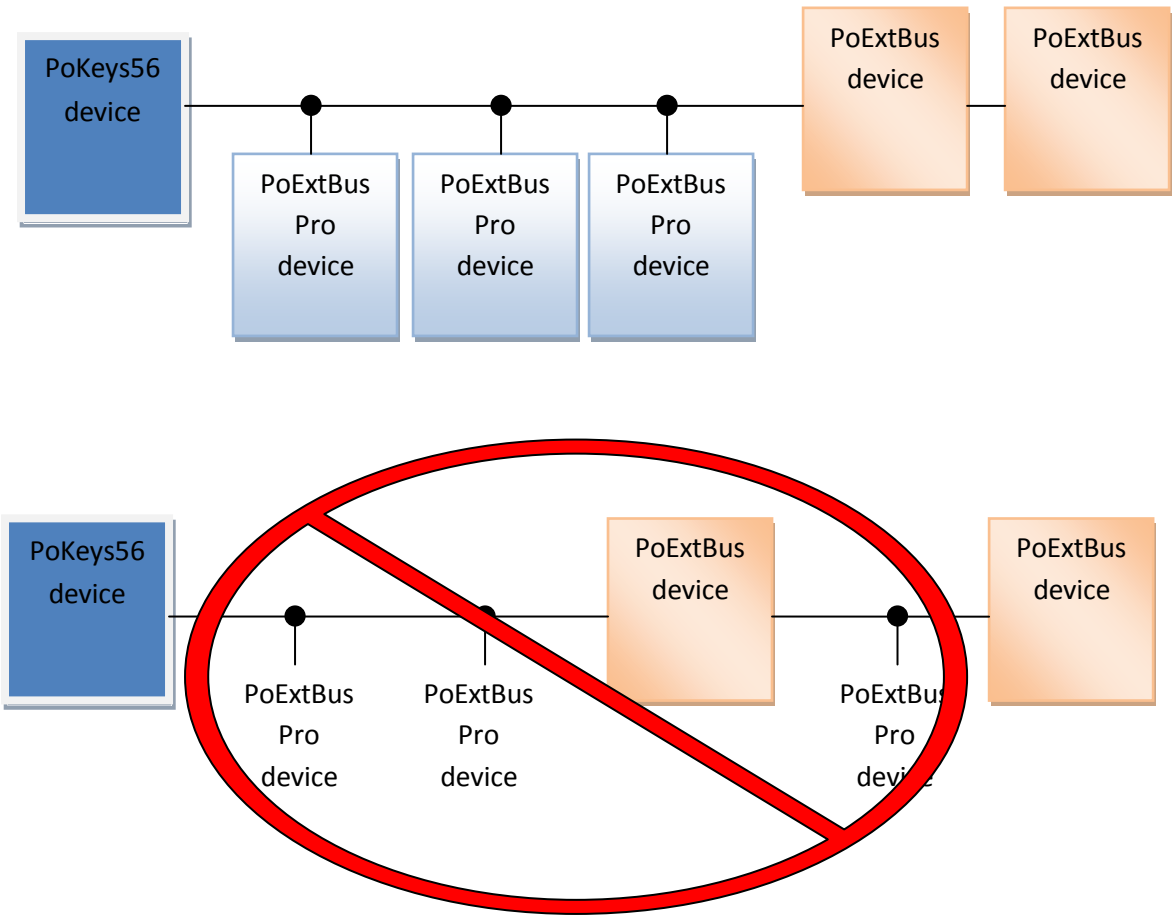
The device can be virually mapped to PoKeys matrix keyboard. Third party software can set the status of LEDs under the keys, read the light sensor that measure the amount of light in the environment and set the intensity of the LEDs.

To map the PoEB KB to PoKeys matrix keyboard, initialize the PoExtBus Pro bus as described above in 'Adding new devices', select the keyboard in the device list and check the checkbox 'Enable mapping to matrix keyboard'.

PoExtBus Pro/PoExtBus connection example

Pay attention to the following:

- **PoExtBus Pro device can be connected only to PoKeys board or other PoExtBus Pro device**
- **PoExtBus device can be connected to either PoKeys board, PoExtBus Pro or PoExtBus device.**
- **There should be not PoExtBus device connected between any PoExtBus Pro devices and PoKeys board**



Changing User ID number

Users can freely assign their own User ID number that represents a specific PoKeys device (enables distinguishing between different PoKeys devices in case there is more than one connected to a single host PC). To change the User ID number, go to ‘Device’ > ‘Change user ID’ menu. Simply enter any number between 0 and 255, and click the 'Change user ID' button.

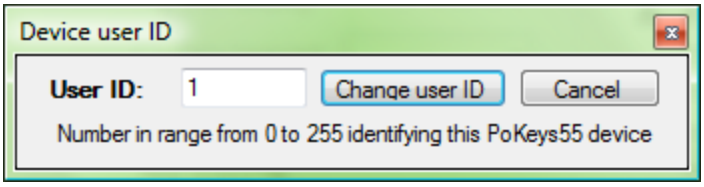


Figure 19: Device user ID dialog

Saving current configuration to file

To save the current configuration to a file, go to ‘File’ > 'Save' menu and select a new filename. To reload a saved configuration from a file, go to ‘File’ > ‘Open’ menu and select the appropriate file. To transfer new settings to the device, click on the ‘Save to device’ button.

Example: starting a program on Windows using PoKeys55 device

On a Windows operating system, users can assign a custom shortcut key to any program shortcut. Find the shortcut and then right click on it to show the context menu (Step 1). Select Properties (Step 2), and under the Shortcut tab (Step 3), click on the ‘Shortcut key’ text box. Proceed by typing in a combination that you wish to assign to a particular program (Step 4). Next, open the PoKeys application and connect to the desired PoKeys device. Click on the pin that will function as a launch trigger for your application (Step 5). Under Key mapping, select the same keyboard combination that you assigned to the program shortcut (Step 6). Click on the ‘Send to device’ button (Step 7) to transfer settings to the device. This will activate the new shortcut.

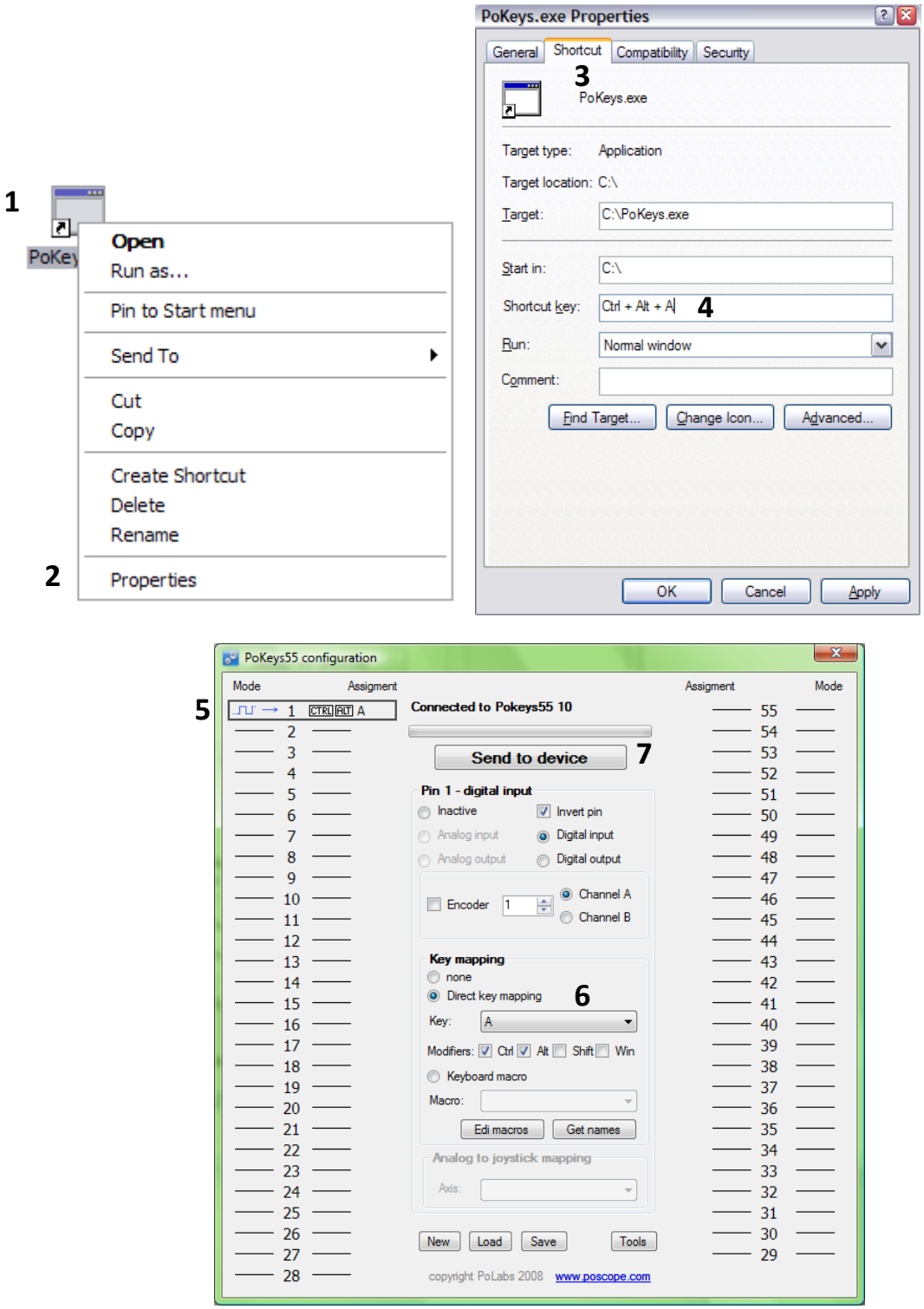


Figure 20: Setting up PoKeys device

7. PoKeys56E

PoKeys56E is a network type of PoKeys device. The device can be connected to Ethernet 10/100 network with standard RJ-45 cable. Battery holder is for future upgrades only and no battery should be inserted into the holder.

By default, the device is set to use the DHCP functionality of the network router. User can later turn on or off the DHCP support. If DHCP is not needed/wanted, fixed IP address can be defined for the device. To set the network settings of the device, go to Device menu and click Change network address...

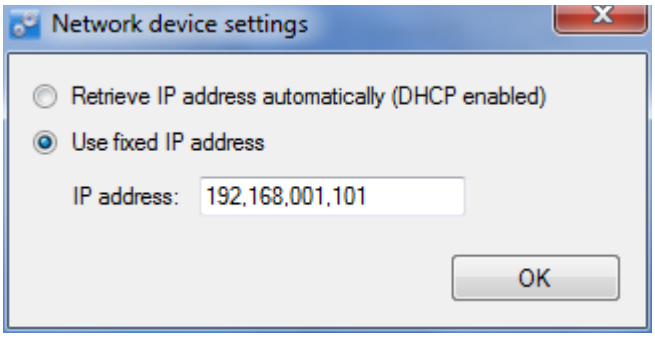


Figure 21: Device network settings

The device communicates using TCP and UDP port of 20055. **Please ensure the firewall settings allow communication with this port. Also, please make sure that your network card (which you have connected PoKeys56E to) has an IP address asigned with the subnet mask 255.255.255.0 (check it in IPv4 settings in your system).**

Connecting to PoKeys56E device for the first time

1. Connect PoKeys56E with your Ethernet network using the RJ-45 cable
2. Connect power supply for PoKeys56E board
    a. If your network uses DHCP to assign IP addresses to each device, PoKeys56E will be assigned a new IP address automatically
    b. If your network uses fixed IP addresses, PoKeys56E device will wait for discovery packet from the PoKeys configuration software (during this time, LED will blink). In this process, temporary IP address from the same subnet will be assigned to PoKeys56E board.
3. Open PoKeys configuration software and wait for devices to be detected
4. If different network configuration is needed, select device and click on Configure button to select new settings.
5. Click on Connect button to connect to PoKeys56E device and start configuring

Device discovery

If device IP address is set as fixed, use can connect directly to PoKeys56E device. Otherwise, UDP discovery packet should be send as UDP broadcast packet (for details, please see the protocol specification document). All PoKeys56E devices that receive this packet, respond with their current IP address and serial number.

Default settings

DHCP:	**enabled**
Port:	20055
Security:	Full access

After receiveing the UDP discovery packet and if the DHCP server is not available, PoKeys56E will use the temporary address of x.x.x.250, where x.x.x is the subnet address (with 255.255.255.0 subnet mask). This enables user to reconfigure the device with the proper IP address. To do this, click on the Configure button.

Connecting to device in other network

When the device is not detected automatically (either there is a firewall blocking the UDP broadcast messages or the device is not in the same network as a computer), custom IP address of the device can be entered by clicking on the 'Network settings... ' button. The following dialog appears.

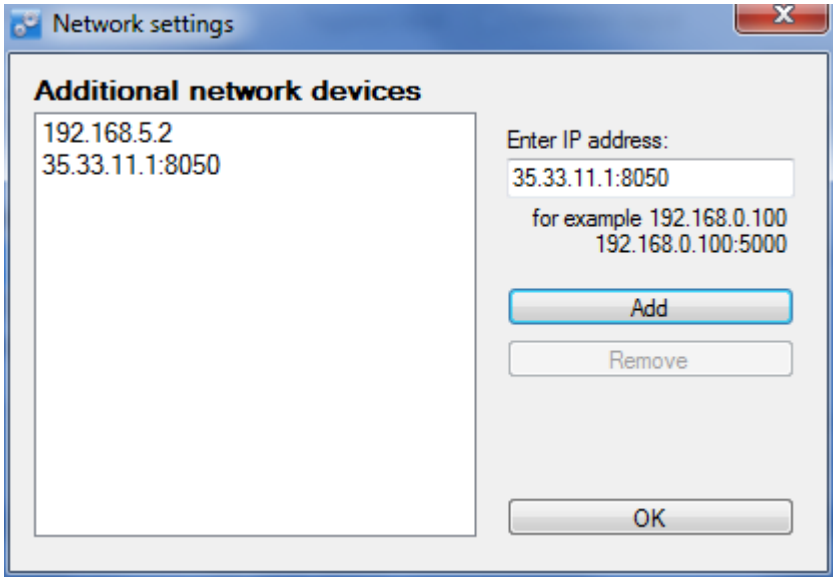


Figure 22: Additional network settings

IP address of the device can be entered in the text box on the right and added to the list by clicking the button 'Add'. The list of additional devices is saved on application exit.

Security

Due to exposed nature of a network device, an authentication mechanism was implemented in PoKeys56E that allows three levels of access rights:

- Full access (default): the device is fully accessible from the network
- Read-only access: unauthorized users are allowed only to fetch data from the device, while an authenticated users can acccess all functions of the device
- Full lock: unauthorised users can not neither read or write to the device. A user password is required to unlock access.

The security is set up in PoKeys configuration software – on the Device menu, click Set device security... The password can contain any character and can be up to 32 characters long.

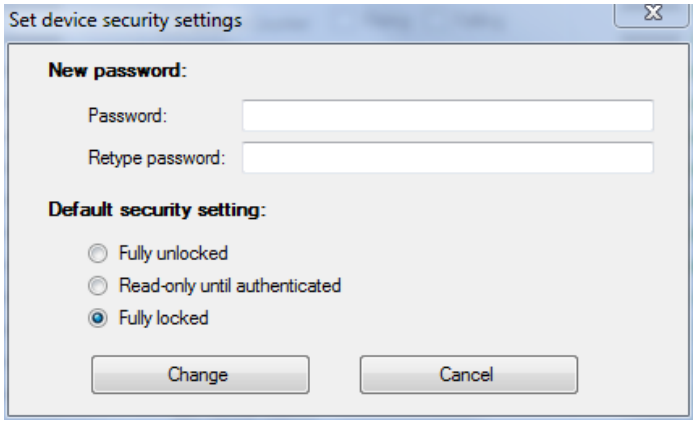


Figure 23: Device security settings window

Modbus support

PoKeys56E supports slave (server) operation of Modbus TCP communication protocol. Modbus TCP compatible devices on the network can read the values from the device and set the outputs. To elevate the security, user can define which peripherals are accessible via Modbus TCP.

Modbus TCP uses TCP protocol on port 502 (default), which can be changed in Modbus settings (accessible from the menu Device – Modbus configuration...). The Modbus TCP connection is disconnected after 3 seconds of inactivity (this default value can be changed in the Modbus settings).

Discrete inputs/outputs

Supported operations:
0x01: Read coils
0x02: Read discrete input
0x05: Write single coil
0x0F: Write multiple coils

Address (0-based)	Access (R – Read, W – Write)	Description
0-54	R/W	55 pin inputs/outputs
1000-1127	R	Matrix keyboard inputs
1200-1327	R	I2C Matrix keyboard inputs
1400-1527	R/W	I2C Matrix keyboard LED
1600-1727	R/W	LED matrix
2000-2079	R/W	PoExtBus
2000		Device 10 – Output H
2001		Device 10 – Output G
...		
2079		Device 1 – Output A

Registers

Supported operations:
0x03: Read holding register
0x04: Read input register
0x06: Write single register
0x10: Write multiple registers

Address (0-based)	Access (R – Read, W – Write)	Description
10-16	R	Analog inputs
20-45	RW	Encoder counter values (lower 16-bit)
100-154	RW	Digital counter values
200-213	RW	PWM
200,201		PWM period (MSB first)
202,203		PWM duty1 (MSB first) – pin 22
…		
212,213		PWM duty6 (MSB first) – pin 17
300-304	RW	PoExtBus
500-579	RW	LCD buffer
600	R	Tick counter (lower 16-bit)

PoExtBus channel mapping:

Address (0-based)	Register description															
300	A	B	C	D	E	F	G	H	A	B	C	D	E	F	G	H
	Device 10								**Device 9**							
301	A	B	C	D	E	F	G	H	A	B	C	D	E	F	G	H
	Device 8								**Device 7**							
302	A	B	C	D	E	F	G	H	A	B	C	D	E	F	G	H
	Device 6								**Device 5**							
303	A	B	C	D	E	F	G	H	A	B	C	D	E	F	G	H
	Device 4								**Device 3**							
304	A	B	C	D	E	F	G	H	A	B	C	D	E	F	G	H
	Device 2								**Device 1**							

Where

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	**Modbus word bit**
A	B	C	D	E	F	G	H	A	B	C	D	E	F	G	H	PoExtBus device bit mapping
Device 10								**Device 9**								

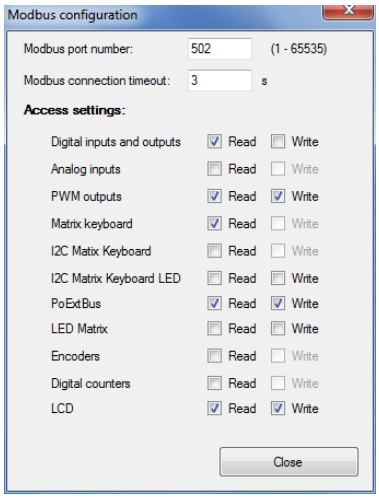


Figure 24: Modbus configuration

Web interface

PoKeys56E devices can be monitored through the simple web interface. To open the interface, use your internet browser and type in the IP address of the PoKeys56E board. The following page will be displayed:

The status of the input pins, encoder values and analog inputs is continuously and dynamically refreshed.

Developers can access status of the inputs by fetching the file /devStat.xml from PoKeys56E board.

Connecting to PoKeys56E across the internet

To access the PoKeys56E from the internet, connect the PoKeys56 board to your router and configure the following port forwarding (consult router manual for instructions on port forwarding):

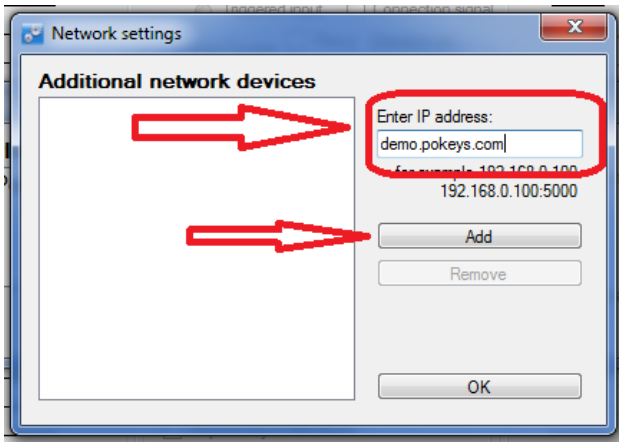
- For the WEB interface, forward the port 80
- For the use of the PoKeys software, forward the port 20055

Using PoKeys software with PoKeys56E device across the internet

Suppose that the PoKeys56E is at the *demo.pokeys.com*.

Open PoKeys software and click on 'Network settings'.

The following dialog will appear. Enter the address of the PoKeys56 board in the field on the right and click on 'Add'.



Then click on 'OK'. If the connection was established successfully, PoKeys56E will be shown in the list of the devices.

8. Support for additional protocols

I²C protocol

The I2C bus was designed by Philips in the early '80s to allow easy communication between components which reside on the same circuit board. Philips Semiconductors migrated to NXP in 2006. The name I2C translates into "Inter IC". Sometimes the bus is called IIC or I²C bus.

PoKeys56E devices support communication with I²C slave devices, connected to the PoExtBus connector. As both I²C and PoExtBus use the same connector, PoExtBus functions are turned off for the time when I²C is activated (PoExtBus functionality can be moved to pins 35,36,37 as with PoKeys55 devices).

Marking the pin closer to the bottom of the board (the oposite side of either Ethernet or USB connector) as pin 1, the I²C devices should be connected as follows:

Pin 1	**Power supply 3.3V**
Pin 2	Ground
Pin 3	Serial data
Pin 4	
Pin 5	Serial clock

Protocol can be tested via PoKeys configuration software. Click on Peripherals > I2C bus test... The following dialog appears.

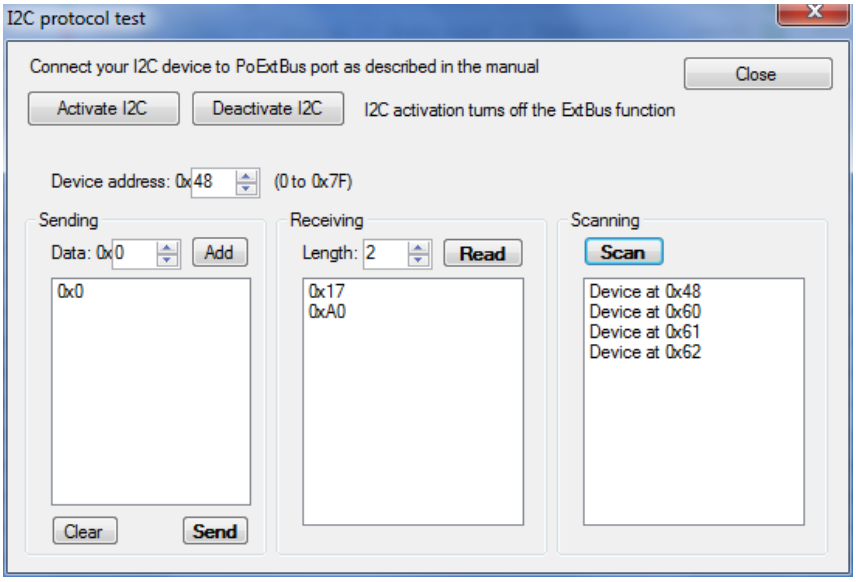


Figure 25: I²C protocol test dialog

1-Wire protocol

1-Wire is a device communications bus system designed by Dallas Semiconductor Corp. that provides low-speed data, signaling, and power over a single signal.[1] 1-Wire is similar in concept to I²C, but with lower data rates and longer range. It is typically used to communicate with small inexpensive devices such as digital thermometers and weather instruments.

PoKeys56E devices support communication with 1-Wire slave devices (without parasitic power supply), connected to the pin 55 with external pull-up resistor (of approximately 5 kΩ).

Protocol can be tested via PoKeys configuration software. Click on Peripherals > 1-Wire bus test... The following dialog appears.

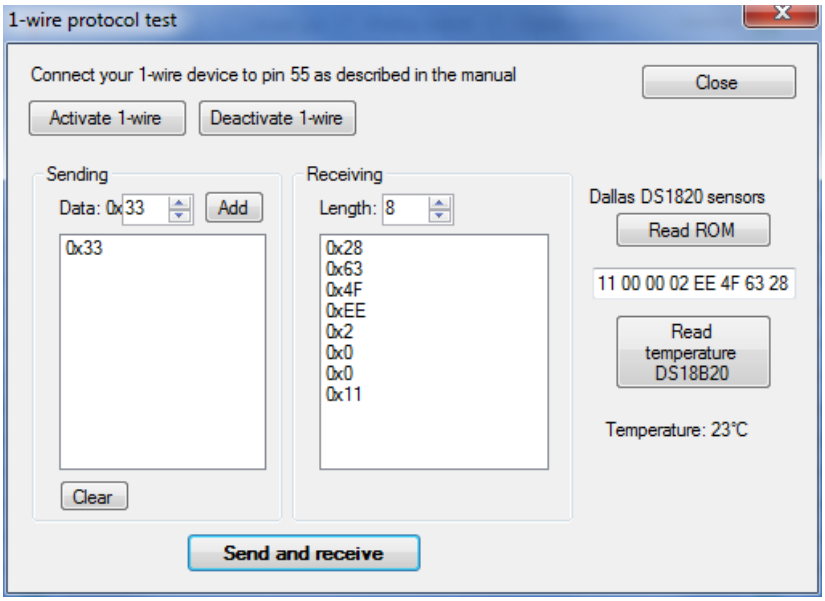


Figure 26: 1-Wire protocol test dialog

9. Communicating with the device using console (PoKeys55 devices only)

In the software bundle included with the PoKeys device, there is also a console interface application, which enables command-line style communication with the device (figure 6). To start using the console application, go Start>Run…, type cmd and press Enter. Navigate to the folder, where PoKeysConsole.exe is located (usually C:\Program Files\PoKeys\).

Supported operations

Enumerate PoKeys55 devices

Command line: PoKeysConsole.exe -e

Description: Enumerates and prints out all the detected PoKeys55 devices with their User IDs.

Get details of the specific PoKeys55 device

Command line: PoKeysConsole.exe –d<user ID>

Description: Prints out the detailed description of the device, i.e. the device's serial number, firmware version and User ID.

Example: PoKeysConsole.exe –d1

Connect to PoKeys55 device

Command line: PoKeysConsole.exe –c<user ID>

Description: Before any operation can be executed, host software must connect to PoKeys55 device, using Connect to PoKeys55 device operation.

Example: PoKeysConsole.exe –c1

Save current configuration to flash memory

Command line: PoKeysConsole.exe –w

Description: After the settings have been changed, they need to be sent to the device. This is accomplished with via the Save configuration operation. 'Connect to PoKeys55 device' operation must be executed before this operation!

Example: PoKeysConsole.exe –c1 –w

Get current pin setting

Command line: PoKeysConsole.exe –g<pin ID>

Description: Prints out the current pin setting. If the pin ID parameter is omitted, settings for all the pins are printed out.

Example: PoKeysConsole.exe –c1 –g10

Set pin setting

Command line: PoKeysConsole.exe –s<pin ID>,<pin function>,+/-

Description: Enables the desired function on the selected pin.

Pin function	Value
Inactive	0
Digital input	2
Digital output	4
Analog input	8
Analog output	16

The last parameter is used to define polarity of digital input and output pins. It must be either + (non-inverted polarity) or - (inverted polarity).

Example: PoKeysConsole.exe –c1 –s10,2,-

Get digital input value

Command line: PoKeysConsole.exe –i<pin ID>

Description: Reads and prints out current digital input value on selected pin.

Example: PoKeysConsole.exe –c1 –i10

Set digital output value

Command line: PoKeysConsole.exe –o<pin ID>,0/1

Description: Sets the digital output to specified value.

Example: PoKeysConsole.exe –c1 –o11,1

Get analog input value

Command line: PoKeysConsole.exe –a<pin ID>

Description: Reads and prints out current analog input value on selected pin. Pin ID must be between 43 and 47, since only these pins support analog to digital conversion.

Example: PoKeysConsole.exe –c1 –a43

Set analog output value

Command line: PoKeysConsole.exe –b<pin ID>,value

Description: Sets the digital output to specified value. Value can be any number between 0 (0 V) and 1023 (3.3 V).

Example: PoKeysConsole.exe –c1 –b43,50

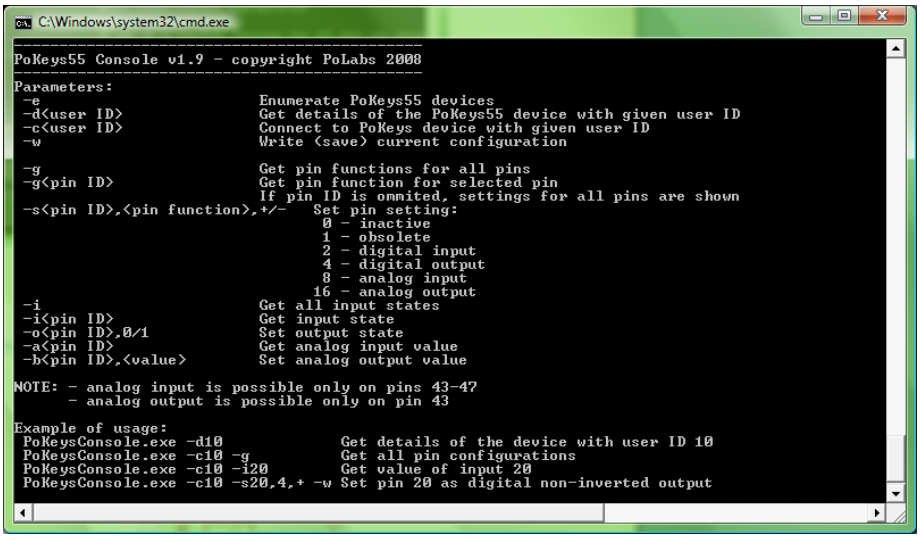
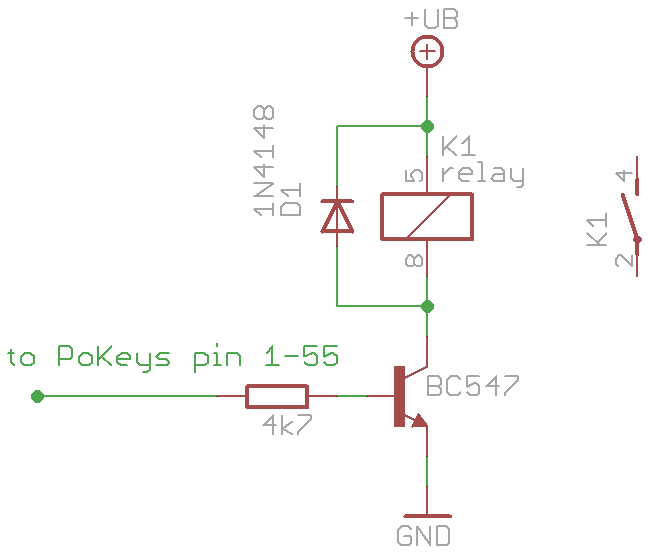

C:\Windows\system32\cmd.exe
PoKeys55 Console v1.9 - copyright PoLabs 2008
Parameters:
-e Enumerate PoKeys55 devices
-d<user ID> Get details of the PoKeys55 device with given user ID
-c<user ID> Connect to PoKeys device with given user ID
-w Write (save) current configuration
-g Get pin functions for all pins
-g<pin ID> Get pin function for selected pin
If pin ID is ommited, settings for all pins are shown
-s<pin ID>,<pin function>,+/- Set pin setting:
0 - inactive
1 - obsolete
2 - digital input
4 - digital output
8 - analog input
16 - analog output
-i Get all input states
-i<pin ID> Get input state
-o<pin ID>,0/1 Set output state
-a<pin ID> Get analog input value
-b<pin ID>,<value> Set analog output value
NOTE: - analog input is possible only on pins 43-47
- analog output is possible only on pin 43
Example of usage:
PoKeysConsole.exe -d10 Get details of the device with user ID 10
PoKeysConsole.exe -c10 -g Get all pin configurations
PoKeysConsole.exe -c10 -i20 Get value of input 20
PoKeysConsole.exe -c10 -s20,4,+ -w Set pin 20 as digital non-inverted output

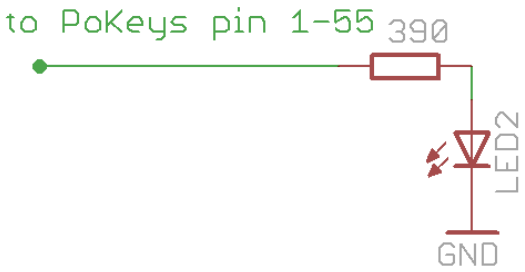
Figure 27: Console application window

10. Connecting common peripherals to PoKeys devices

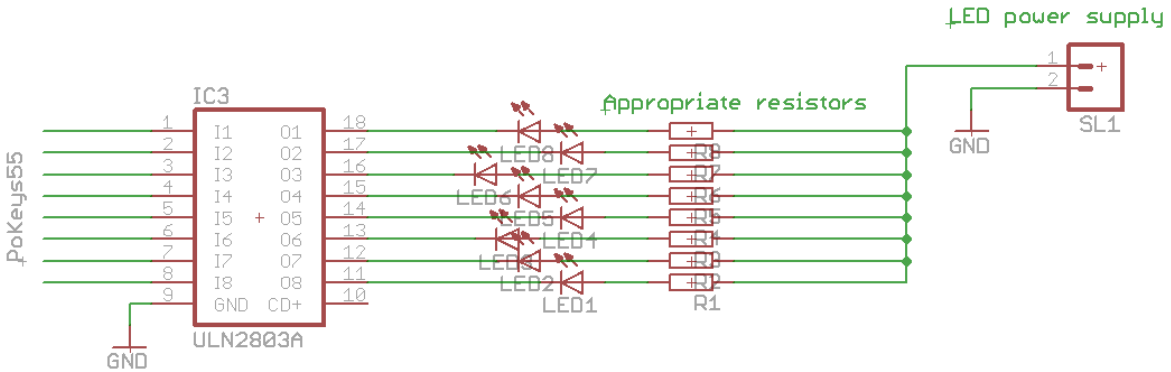
1. Relays



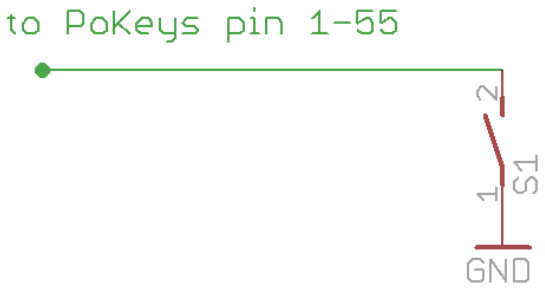
2. LEDs



3. High-current LEDs



4. *Switches*

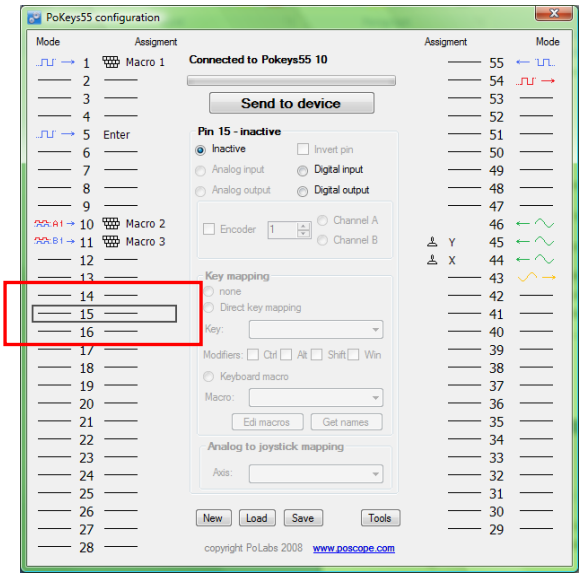


Example: Setting up key mapping

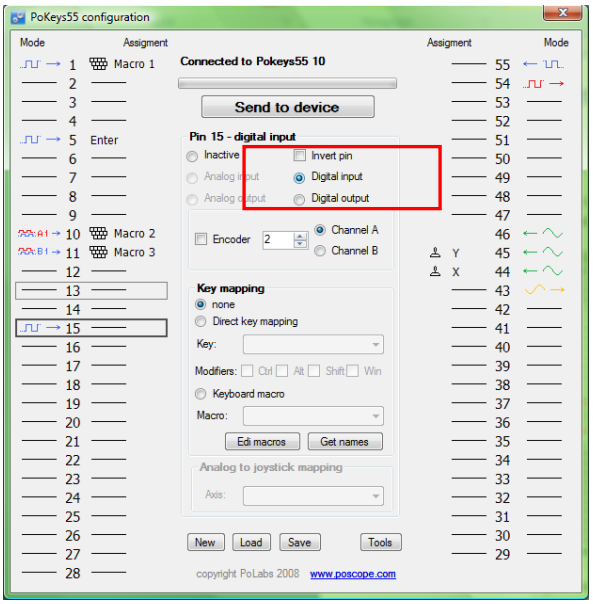
This example shows how easy is to set up a digital input pin for direct key mapping

We will set up a Shift-Escape combination for pin 15.

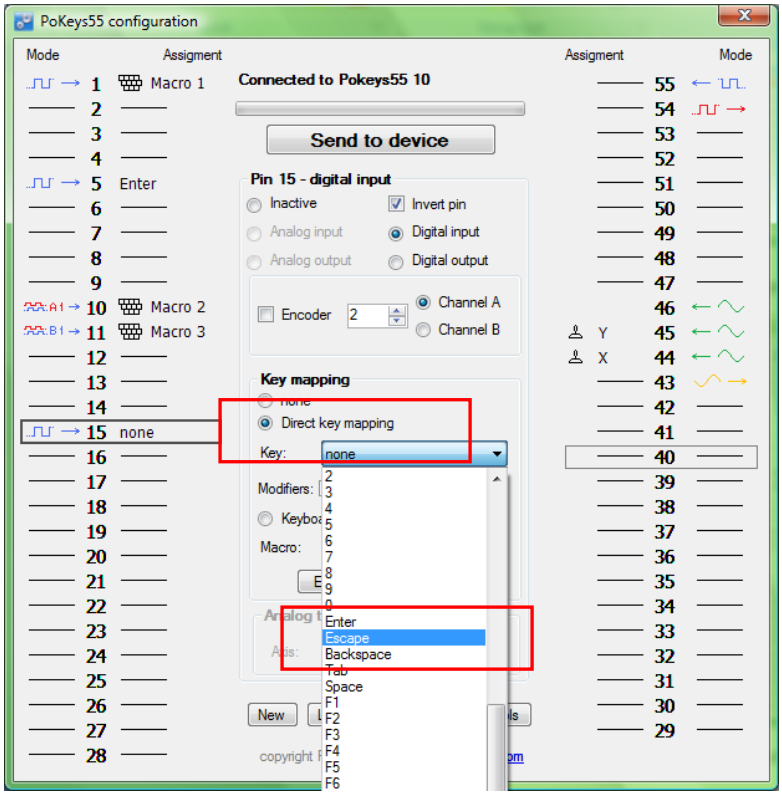
1. Connect a switch to your PoKeys device as shown above
2. Open PoKeys configuration application
3. Select your PoKeys device from drop-down box and click ‘Connect’ button
4. Wait the application to load current configuration from PoKeys device
5. Click the same pin number as you connected a switch to (in this example pin 15)



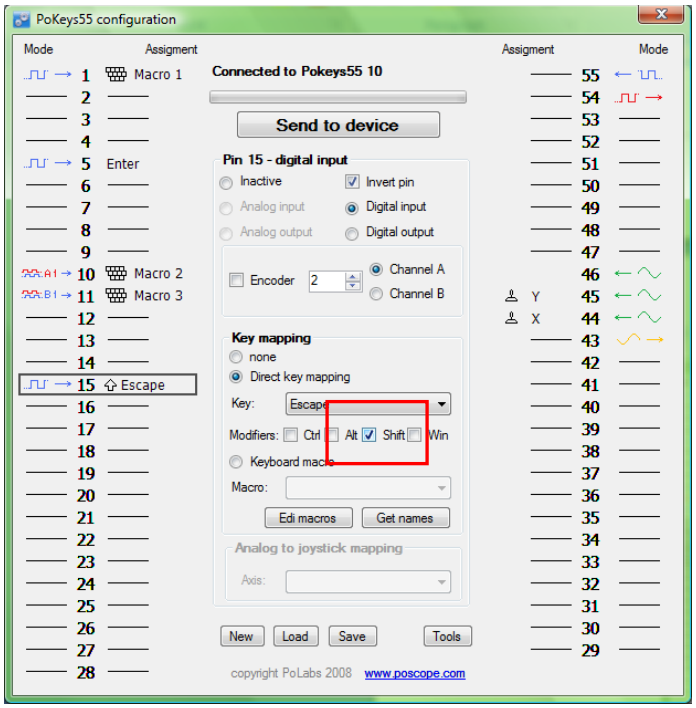
6. Set this pin as digital input



7. Select ‘Direct key mapping’ and from drop-down box select Escape

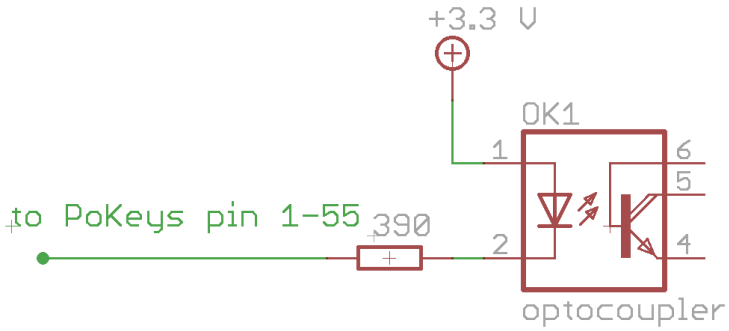


8. Click on the ‘Shift’ checkbox to enable Shift modifier

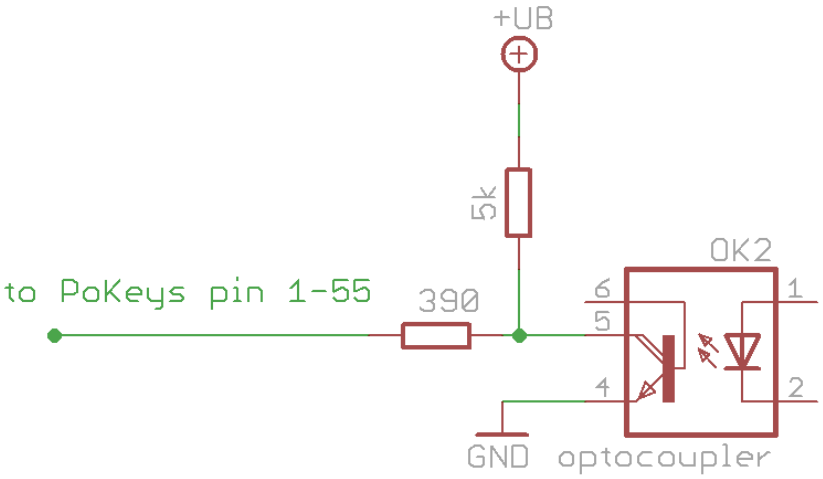


9. Send configuration to device by clicking ‘Send to device button’.

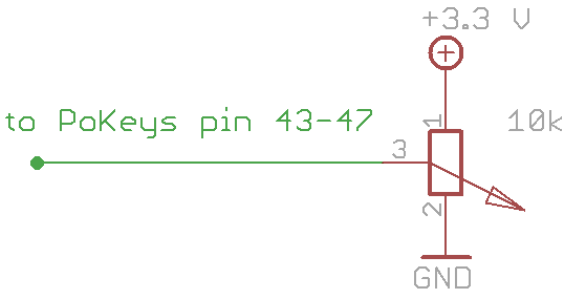
5. *Optocoupled digital output*



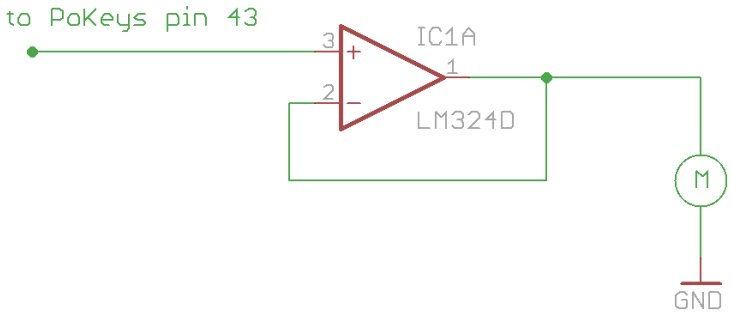
6. *Optocoupled digital input*



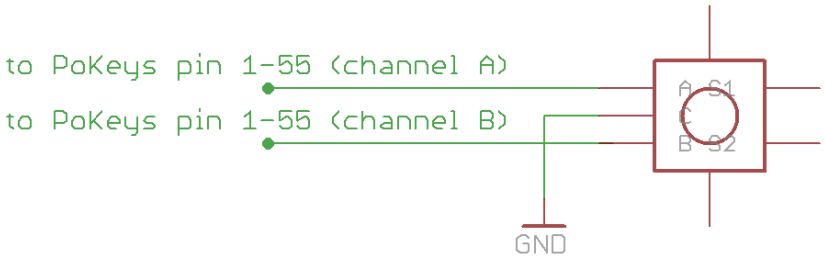
7. Potentiometers (variable resistors)



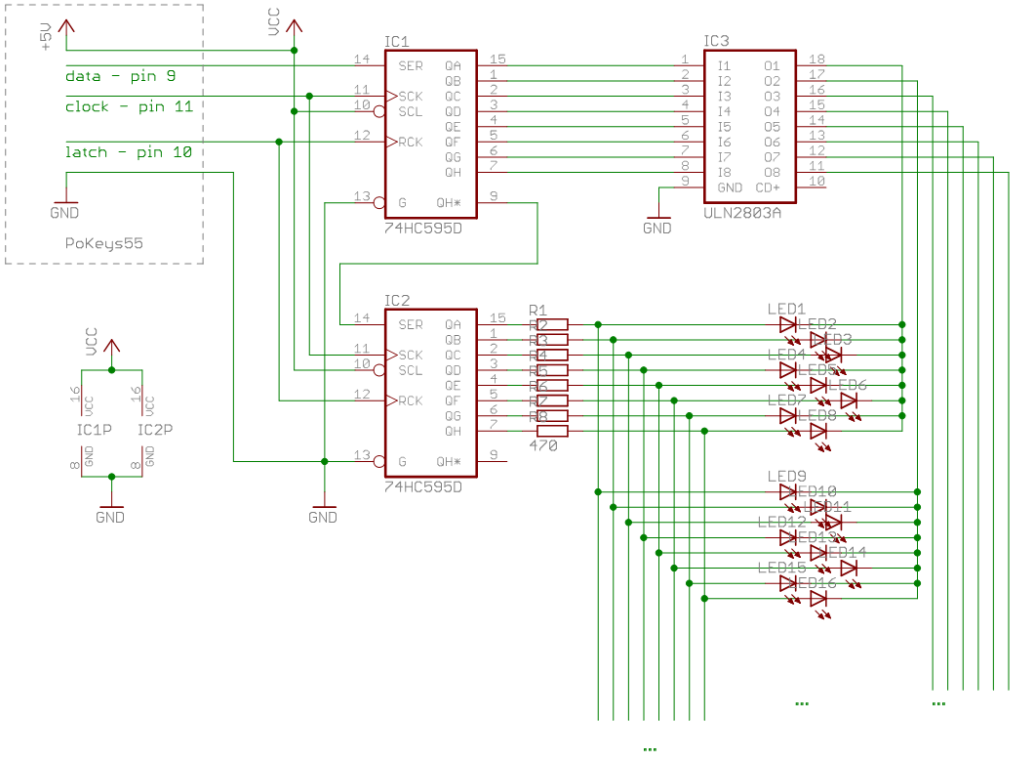
8. Linear motor control



9. Rotational encoder switch



10. LED displays/arrays


+5V
VCC
data - pin 9
clock - pin 11
latch - pin 10
GND
PoKeys55
IC1
SER
SCK
SCL
RCK
G
QA
QB
QC
QD
QE
QF
QG
QH
QH*
74HC595D
IC2
74HC595D
IC3
I1
I2
I3
I4
I5
I6
I7
I8
GND
O1
O2
O3
O4
O5
O6
O7
O8
CD+
ULN2803A
R1
470
LED1
LED2
LED3
LED4
LED5
LED6
LED7
LED8
LED9
LED10
LED11
LED12
LED13
LED14
LED15
LED16
IC1P
IC2P

11. Quick resetting the device configuration

If configuration editor cannot be used to reconfigure the device because of endless key presses from the device, simple reset procedure should be executed.

1. Disconnect PoKeys55 device from USB
2. Find pin labeled ‘Reset’ on the PoKeys device (otherwise use pin numbered 54)
3. Short this pin to ground (GND) and reconnect the PoKeys device to USB
4. Green light should start flashing rapidly
5. Wait approximately 10s, the light will start flashing slowly
6. PoKeys device will connect in recovery mode
7. Open PoKeys configuration application
8. PoKeys configuration application should detect PoKeys device in recovery mode and present you with an option ‘Clear settings’. By clicking this button and confirming your decision on the next dialog, settings will be erased.
9. Disconnect ‘Reset’ pin from ground and unplug PoKeys device.
10. To ensure, the device is properly cleared, replug PoKeys device, connect to it and execute Device > Clear settings in device.

PoKeys configuration software is backing up current configuration state (except keyboard macro sequences) on each connection start.

These configuration files can be found in the local application folder (system folder – usually c:\Documents and settings\{username}\Local Settings\Application Data\PoKeys55\ on Windows 2000, XP or C:\Users\{username}\AppData\Local\PoKeys55\), named backup1.pkc, backup2.pkc and backup3.pkc with backup3.pkc being the oldest configuration.

12. Frequently asked questions

What software must be installed to operate the device?

On first use or when reconfiguring the device, the supplied software must be installed. There are no device drivers needed. They are already supplied with your operating system. Once the device has been configured, the settings are stored on-board. Device can then be freely used on any machine (see requirements for USB HID device driver enabled operating system) without any additional installation.

I misconfigured the device. Now the device starts pressing virtual keys before I can do anything. What can I do?

If you misconfigured the device in such a way that configuration utility cannot be used to repair the configuration, see the section ‘Quick resetting the device configuration’ in this manual.

How do I connect switch/relay/LED/… to PoKeys device?

Please see the section ‘Connection common peripherals to PoKeys device' in this manual.

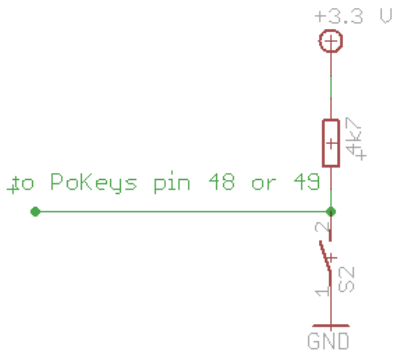
I have two (or more) PoKeys devices connected on one system and cannot differentiate the devices to set the configurations.

It is advised that the users assign different UserID numbers to each of the device connected to a system. Please see the section ’0

Changing User ID number' of this manual.

It appears that pins 48 and 49 are floating. What should I do?

Due to device design, pins 48 and 49 should be equipped with external 5-10 kΩ pull-up resistor as shown bellow.



Note: on PoKeys55 boards with serial number above 11500, this problem is removed.

I have an encoder connected to the PoKeys device and when I read the raw data, the number climbs to 255 and then loops back around. Shouldn't the number continue to increase (or decrease depending on which way I turn the shaft)?

RAW data is an 8-bit number and this is normal behavior and is named overflow. If you wish to use greater range, you should extend it in your program. Check the RAW data register periodically and calculate differences between states of the register at these periodic times in 8-bit space. Then use the difference calculated to increment or decrement your own 16-bit, 32-bit or wider register.

I have connected a switch to pin 4 and now PoKeys55 is not recognized by the computer anymore.

You must have connected normally-closed switch to pin 4 and therefore connected pin 4 to ground. At boot (connecting PoKeys55 to USB) this means that PoKeys55 is entering system boot and therefore cannot be used from the computer. Please use another pin for normally-closed switches.
Note: on PoKeys55 boards with serial number above 11500, this problem is removed.

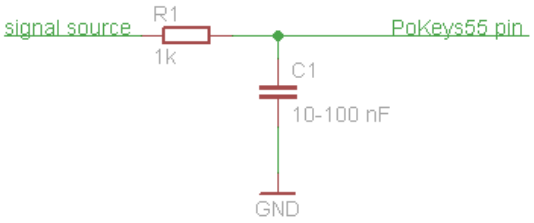
I have connected a switch to pin 54 and now PoKeys55 is connecting in recovery mode on every boot.

You must have connected normally-closed switch to pin 54 and therefore connected pin 4 to ground. At boot (connecting PoKeys55 to USB) this means that PoKeys device is entering recovery mode and therefore cannot be used from the third-party software. Please use another pin for normally-closed switches. Also see chapter 9: Quick resetting the device configuration.

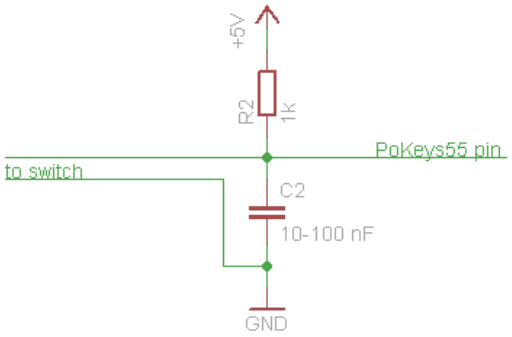
There is spontaneous triggering of some of the pins. What is wrong?

You might have connected long cable from the PoKeys board to the switch. If this cable crosses any power cables, it can trigger a false signal on PoKeys board input due to interference or coupling. Use twisted pair wires if possible. If this not help and cables cannot be routed elsewhere, use shielded cable.

If cables are routed inside electrically 'dirty' environment, use simple RC filter on those signals as illustrated below.



If connecting switches to PoKeys device with long cables, use the schematics below. When using matrix keyboard connected to PoKeys device with long cables, it is also advised to use the schematics below for digital input pins. This filter should not be used on digital output pins!



My problem is that when I start the program, everything works good for about 3 to 5 seconds, then the CPU "lock up" and the only way to recover is to unplug the PoKeys device.

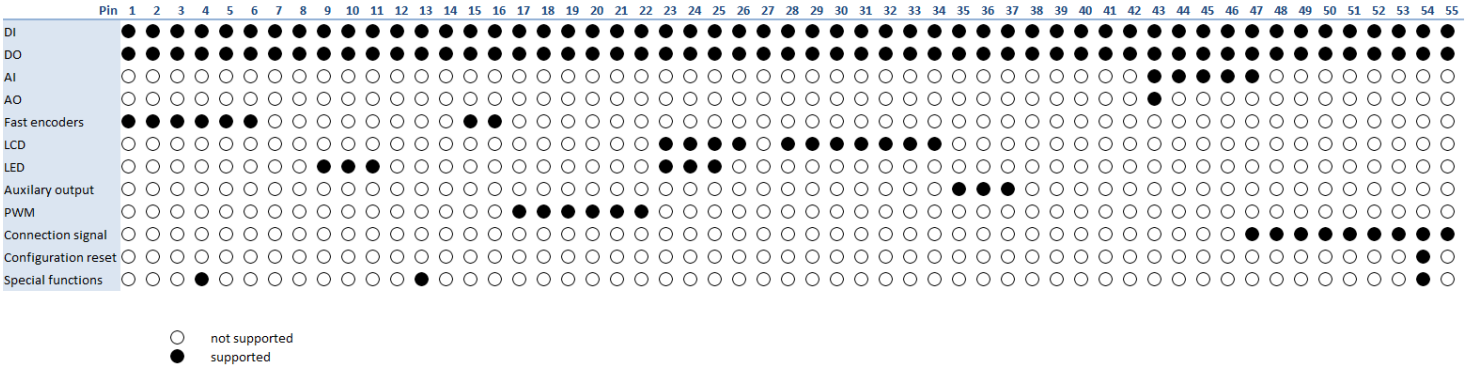
The problem occurs because you are writing 'dirty' code. You create object every time you need to use it in a loop, but you forget do properly dispose it. Best way to use PoKeys DLL in an application that read or writes data in a loop, is to create a global object and initialize it once at the start of application, and use its functions to read or write in a loop. This way the communication is much faster.

If you are using Visual Basic development environment, add a reference to PoKeysDevice DLL and use object browser to find proper declaration. Via object browser you can also access the list of all supported functions, which will also be used by Intelli sense in editor.

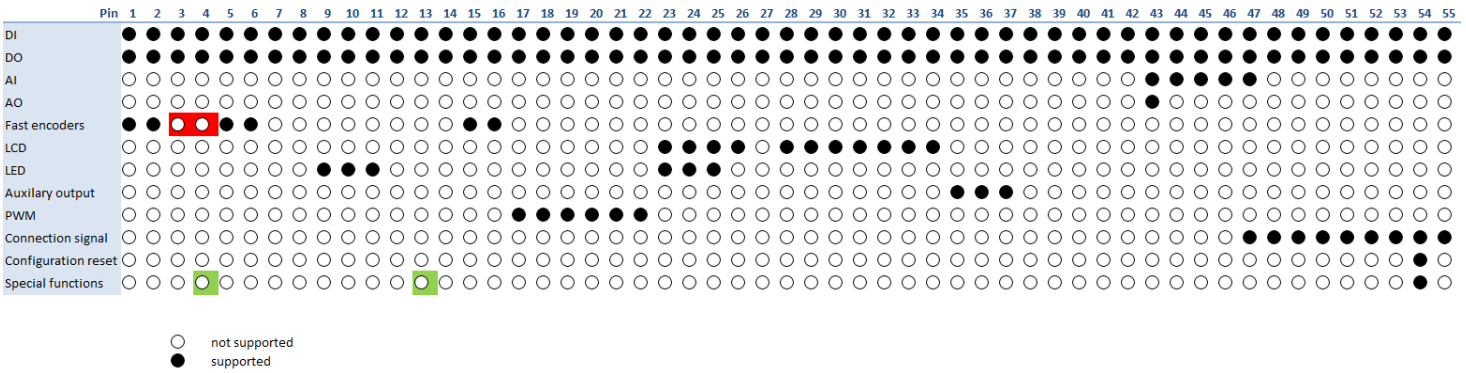
What can I connect to each pin of the PoKeys device?

Please take a look at the table below.

For PoKeys55 devices with serial numbers between 10000 and 11499:

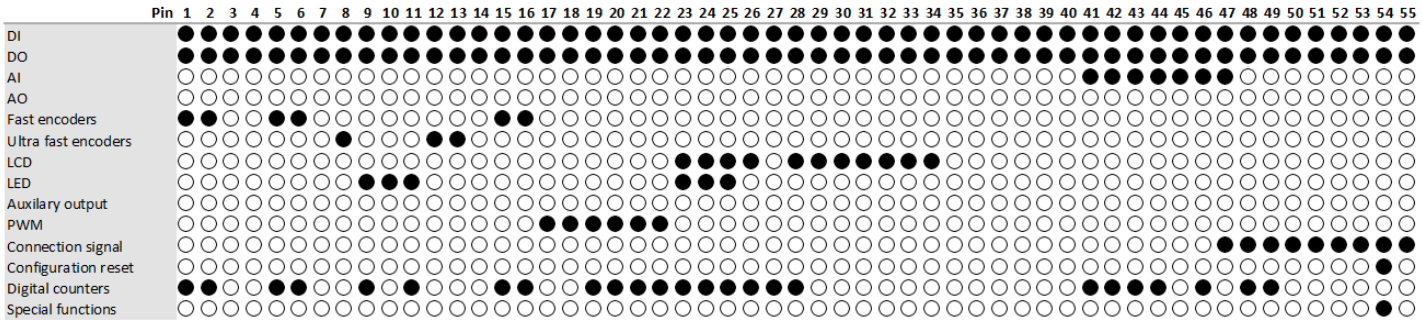

Pin 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55
DI
DO
AI
AO
Fast encoders
LCD
LED
Auxilary output
PWM
Connection signal
Configuration reset
Special functions
not supported
supported

For PoKeys55 devices with serial numbers above 11500:


Pin 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55
DI
DO
AI
AO
Fast encoders
LCD
LED
Auxilary output
PWM
Connection signal
Configuration reset
Special functions
not supported
supported

Due to some hardware changes, some features had to be moved between pins, some problematic pins were removed.

For PoKeys56E devices:


Pin 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55
DI
DO
AI
AO
Fast encoders
Ultra fast encoders
LCD
LED
Auxilary output
PWM
Connection signal
Configuration reset
Digital counters
Special functions

13. PoKeys library functions

EnumerateDevices (for USB-based PoKeys devices)

Enumerate the PoKeys devices and return number of found PoKeys devices.

```
int EnumerateDevices()
```

Arguments:

none

Remarks:

This function must be called on every class initialization

ConnectToDevice (for USB-based PoKeys devices)

Connect to the device with the index specified.

```
bool ConnectToDevice(int deviceIndex)
```

Arguments:

deviceIndex

Index of the PoKeys device

Remarks:

Index is not UserID of the PoKeys device and therefore can change if more than one PoKeys device is used at a time. Function returns True if connection is established or False if there were errors.

DisconnectDevice

Terminate the connection with the device.

```
void DisconnectDevice()
```

Arguments:

none

Remarks:

This function should be called before class disposal or changing of the device.

StartEthernetDiscovery

Starts the automatic discovery of PoKeys56E devices.

```
void StartEthernetDiscovery ()
```

Arguments:

none

Remarks:

See example section for usage example

StopEthernetDiscovery

Stops the automatic discovery of PoKeys56E devices.

```
void StopEthernetDiscovery ()
```

Arguments:

none

Remarks:

See example section for usage example

GetNumberOfDetectedNetworkDevices

Returns the number of detected PoKeys56E devices.

```
int GetNumberOfDetectedNetworkDevices()
```

Arguments:

none

Remarks:

See example section for usage example

ConnectToNetworkDevice

Connects to PoKeys56E device with the specified IPv4 address in format xxx.xxx.xxx.xxx (e.g., 192.168.0.105, 192.168.050.002, etc.). If connection is established successfully, returns True.

```
bool ConnectToNetworkDevice(string address)
```

Arguments:

address

IPv4 address of the selected device

Remarks:

See example section for usage example

GetDeviceID

Retrieve device ID data, i.e. serial number and firmware version.

```
bool GetDeviceID(ref int serialNumber, ref int firmwareVersion, ref int pinNum)
```

Arguments:

serialNumber

Variable in which serial number will be saved to

firmwareVersion

Variable in which firmware version will be saved to

pinNum

Variable in which number of pins will be saved to

Remarks:

It is advised to use GetDeviceIDEx instead of this function. Returns False on error.

GetDeviceIDEx

Retrieve device ID data, i.e. serial number and firmware version.

```
bool GetDeviceIDEx(ref int serialNumber, ref int firmwareVersionMajor, ref int firmwareVersionMinor)
```

Arguments:

serialNumber

Variable in which serial number will be saved to

firmwareVersionMajor

Variable in which firmware major version will be saved to

firmwareVersionMinor

Variable in which firmware minor version will be saved to

Remarks:

Returns False on error.

GetBuildDate

Retrieve firmware build date.

```
bool GetBuildDate(ref string buildDate)
```

Arguments:

buildDate

Variable in which build date will be saved to

Remarks:

Returns False on error.

GetUserID

Retrieve user ID.

```
bool GetUserID(ref byte userID)
```

Arguments:

userID

Variable in which user ID will be saved to

Remarks:

Returns False on error.

SetUserID

Set user ID.

```
bool SetUserID(byte newUserID)
```

Arguments:

newUserID

New user ID

Remarks:

It is advised that each PoKeys device on a system should have its unique user ID. Returns False on error.

SetPinData

Set pin data – pin's function and options.

```
bool SetPinData(byte pinID, byte pinFunction, byte pullUpDownResistor, byte invertPin)
```

Arguments:

pinID

Pin ID is zero-based pin index on the device (output marked as 1 therefore has index 0)

pinFunction

pinFunction has the following structure

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Pin invert	reserved	reserved	A output	A input	D output	D input	reserved

where A stands for analog and D for digital.

pullUpDownResistor

No function at the moment

invertPin

No function at the moment, see bit 7 of pinFunction argument

Remarks:

It is advised to use function SetPinData with only 2 parameters (pinID and pinFunction). Returns False on error.

SetPinData

Set pin data – pin's function and options.

```
bool SetPinData(byte pinID, byte pinFunction)
```

Arguments:

pinID

Pin ID is zero-based pin index on the device (output marked as 1 therefore has index 0)

pinFunction

pinFunction has the following structure

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Pin invert	reserved	reserved	A output	A input	D output	D input	reserved

where A stands for analog and D for digital.

Remarks:

Returns False on error.

GetPinData

Get pin data – pin's function and options.

```
bool GetPinData(byte pinID, ref byte pinFunction, ref byte pullUpDownResistor, ref byte invertPin, ref byte pinPossibleFunctions)
```

Arguments:

pinID

Pin ID is zero-based pin index on the device (output marked as 1 therefore has index 0)

pinFunction

pinFunction has the following structure

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Pin invert	reserved	reserved	A output	A input	D output	D input	reserved

where A stands for analog and D for digital.

pullUpDownResistor

No function at the moment

invertPin

No function at the moment, see bit 7 of pinFunction argument

pinPossibleFunctions

No function at the moment

Remarks:

It is advised to use function GetPinData with only 2 parameters (pinID and pinFunction). Returns False on error.

GetPinData

Get pin data – pin's function and options.

```
bool GetPinData(byte pinID, byte pinFunction)
```

Arguments:

pinID

Pin ID is zero-based pin index on the device (output marked as 1 therefore has index 0)

pinFunction

pinFunction has the following structure

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Pin invert	reserved	reserved	A output	A input	D output	D input	reserved

where A stands for analog and D for digital.

Remarks:

Returns False on error.

GetInput

Get digital input status.

```
bool GetInput(byte pinID, ref bool inputState)
```

Arguments:

pinID

Pin ID is zero-based pin index on the device (output marked as 1 therefore has index 0)

inputState

Variable in which input state will be saved to

Remarks:

Pin must be set as digital input for this command to function properly. Returns False on error.

SetOutput

Set digital output status.

```
bool SetOutput(byte pinID, bool outputState)
```

Arguments:

pinID

Pin ID is zero-based pin index on the device (output marked as 1 therefore has index 0)

outputState

New output state

Remarks:

Pin must be set as digital output before this operation will function properly. On device initialization, pins are NOT set as outputs even if they are configured so. Before using the pins as digital outputs SetPinFunction must be called to set the direction of the pin. This must be repeated on every startup. Returns False on error.

GetAnalogInput

Get analog input status.

```
bool GetAnalogInput(byte pinID, ref [int,byte] inputValue)
```

Arguments:

pinID

Pin ID is zero-based pin index on the device (output marked as 1 therefore has index 0)

inputValue

Variable in which analog value will be saved to. Use byte type for 8-bit analog values and int type for 10-bit analog values

Remarks:

The returned value is between 0 and 255 (8-bit resolution) or 0 and 1023 (10-bit resolution). 0 means 0 V on input, while 255 respectively 1023 means Vdd (approximately 3.3V) on input. Returns False on error.

SetAnalogOutput

Get analog input status.

```
bool SetAnalogOutputValue (byte pinID, [int,byte] value)
```

Arguments:

pinID

Pin ID is zero-based pin index on the device (output marked as 1 therefore has index 0)

value

New 8-bit or 10-bit value for analog output.

Remarks:

The value specified must be between 0 and 255 (8-bit resolution) or 0 and 1023 (10-bit resolution). 0 means 0 V on output, while 255 respectively 1023 means Vdd (approximately 3.3V) on output. Returns False on error.

SaveConfiguration

Saves current device configuration to non-volatile flash memory.

```
bool SaveConfiguration()
```

Arguments:

none

Remarks:

This function takes some time to complete. Is meantime, communication with the device is not possible. Returns False on error.

GetMatrixConfiguration

Get complete matrix keyboard configuration.

```
bool GetMatrixKeyboardConfiguration(ref byte configuration, ref byte width, ref byte height, ref byte[] row_pins, ref byte[] column_pins, ref bool[] macro_mapping, ref byte[] keycodes, ref byte[] keymodifiers);
```

Arguments:

configuration

If bit 0 is set, matrix keyboard is enabled. Other bits are reserved

width, height

Number of columns and rows of the matrix keyboard

row_pins

An array of 8 bytes, each having an index of a pin that is associated with the row. Row pins must be set as digital outputs.

column_pins

An array of 8 bytes, each having an index of a pin that is associated with the column. Column pins must be set as digital inputs.

macro_mapping

An array of 64 boolean values (see below for numbering hint). If the value is set to true, instead of key press simulation, macro is run.

keycodes

An array of 64 byte values (see below for numbering hint). If appropriate macro_mapping value is set to true, each value can contain index of a macro else it contains code of a key.

keymodifiers

An array of 64 byte values (see below for numbering hint). It contains key modifiers.

Keys indexing:

No matter what dimensions the matrix keyboard has, the following scheme is used for keys indexing. A1 is always 0, B1 1, A2 8, ... For example, if user connects a 3x3 matrix keyboard, keys have indexes: 0, 1, 2, 8, 9, 10, 16, 17, 18.

	A	B	C	D	E	F	G	H
1	0	1	2	3	4	5	6	7
2	8	9	10	11	12	13	14	15
3	16	17	18	19	20	21	22	23
4	24	25	26	27	28	29	30	31
5	32	33	...					

Remarks:

Row pins must be set as digital outputs and column pins as digital inputs respectively. Returns False on error.

SetMatrixConfiguration

Set complete matrix keyboard configuration.

```
bool SetMatrixKeyboardConfiguration(ref byte configuration, ref byte width, ref byte height, ref byte[] row_pins, ref byte[] column_pins, ref bool[] macro_mapping, ref byte[] keycodes, ref byte[] keymodifiers);
```

Arguments:

configuration

If bit 0 is set, matrix keyboard is enabled. Other bits are reserved

width, height

Number of columns and rows of the matrix keyboard

row_pins

An array of 8 bytes, each having an index of a pin that is associated with the row.

column_pins

An array of 8 bytes, each having an index of a pin that is associated with the column.

macro_mapping

An array of 64 boolean values (see below for numbering hint). If the value is set to true, instead of key press simulation, macro is run.

keycodes

An array of 64 byte values (see below for numbering hint). If appropriate macro_mapping value is set to true, each value can contain index of a macro else it contains code of a key.

keymodifiers

An array of 64 byte values (see below for numbering hint). It contains key modifiers.

Keys indexing:

No matter what dimensions the matrix keyboard has, the following scheme is used for keys indexing. A1 is always 0, B1 1, A2 8, ... For example, if user connects a 3x3 matrix keyboard, keys have indexes: 0, 1, 2, 8, 9, 10, 16, 17, 18.

	A	B	C	D	E	F	G	H
1	0	1	2	3	4	5	6	7
2	8	9	10	11	12	13	14	15
3	16	17	18	19	20	21	22	23
4	24	25	26	27	28	29	30	31
5	32	33	...					

Remarks:

Returns False on error.

GetPWMOutputs

Get complete PWM outputs configuration

```
bool GetPWMOutputs(ref bool[] channels, ref uint period, ref uint[] duty_values);
```

Arguments:

channels

An array of 6 boolean values, each representing one PWM channel. Channel is enabled if this value is set to true.

period

32-bit PWM period value. PWM module of a PoKeys55 device runs at 12 Mhz, so a value of 12 000 000 produces a period of 1 second. PWM module on PoKeys56E devices runs at 25 MHz, so a value of 25 000 000 produces a period of 1 second.

duty_values

An array of 6 32-bit unsigned integers, each representing the value of PWM duty for each channel. Minimum value is 0, maximum value is the same as period.

Channel to pin mapping:

Channels are mapping according to this table:

Channel	PoKeys pin
0	22
1	21
2	20
3	19
4	18
5	17

Remarks:

When using PWM enabled pins, digital inputs and outputs are inactive. Returns False on error.

SetPWMOutputs

Set complete PWM outputs configuration

```
bool SetPWMOutputs(ref bool[] channels, ref uint period, ref uint[] duty_values);
```

Arguments:

channels

An array of 6 boolean values, each representing one PWM channel. Channel is enabled if this value is set to true.

period

32-bit PWM period value. PWM module of a PoKeys55 device runs at 12 Mhz, so a value of 12 000 000 produces a period of 1 second. PWM module on PoKeys56E devices runs at 25 MHz, so a value of 25 000 000 produces a period of 1 second.

duty_values

An array of 6 32-bit unsigned integers, each representing the value of PWM duty for each channel. Minimum value is 0, maximum value is the same as period.

Channel to pin mapping:

Channels are mapping according to this table:

Channel	PoKeys pin
0	22
1	21
2	20
3	19
4	18
5	17

Remarks:

When using PWM enabled pins, digital inputs and outputs are inactive. Returns False on error.

LCDSetSettings

Set LCD settings.

```
bool LCDSetSettings(byte option, byte rows, byte cols)
```

Arguments:

option

Set to 1 to enable LCD support.

rows

number of rows on LCD module used.

cols

number of columns on LCD module used.

Remarks:

Enabling LCD support does NOT disable pins. Special care should be taken when simultanously using LCD designated pins for LCD display and digital inputs or outputs.

LCDInit

Initializes the LCD module.

```
bool LCDInit()
```

Arguments:

No arguments needed

Remarks:

No remarks.

LCDClear

Clears LCD display.

```
bool LCDClear()
```

Arguments:

No arguments needed

Remarks:

No remarks.

LCDGotoXY

Move cursor to position x (column), y (row)

```
bool LCDGotoXY(byte x, byte y)
```

Arguments:

x

Column

y

Row

Remarks:

x and y are 1-based addresses. Home position therefore is at 1,1.

LCDPutc

Displays one character on LCD module.

```
bool LCDPutc(char character)
```

Arguments:

character

Character code of character to be displayed

Remarks:

Character is displayed from at current position of the cursor.

LCDPrint

Displays string on LCD

```
bool LCDPrint(string LCDText)
```

Arguments:

LCDText

String containing up to 20 characters and \0 termination.

Remarks:

Text is displayed from the current position of the cursor.

LCDSetEntryMode

Set option for 'Entry mode'.

```
bool LCDSetEntryMode(byte CursorMoveDirection, byte DisplayShift)
```

Arguments:

CursorMoveDirection

0 for left-to-right movement, 1 for right-to-left movement

DisplayShift

0 for display shift off, 1 for display shift on

Remarks:

No remarks.

LCDDisplayOnOffControl

Set option for 'On/off control'.

```
bool LCDDisplayOnOffControl(byte DisplayOnOff, byte CursorOnOff, byte CursorBlinkingOnOff)
```

Arguments:

DisplayOnOff

1 for display on.

CursorOnOff

1 for cursor display on.

CursorBlinkingOnOff

1 for cursor blinking on

Remarks:

No remarks.

LCDDefineCustomCharacter

Defines one custom character.

```
bool LCDDefineCustomCharacter(byte CharacterCode, byte[] characterData)
```

Arguments:

Character code

Character code in range from 0 to 7

characterData

8-byte array containing custom character data as specified in LCD module documentation.

Remarks:

No remarks.

AuxilaryBusGetData

Reads PoExtBus enabled setting.

```
bool AuxilaryBusGetData(ref byte auxilaryBusEnabled)
```

Arguments:

auxilaryBusEnabled

PoExtBus enabled status (if 1, PoExtBus is enabled, otherwise disabled)

Remarks:

No remarks.

AuxilaryBusSetData

Sets-up PoExtBus settings and data.

```
bool AuxilaryBusSetData(byte auxilaryBusEnabled, byte[] dataBytes)
```

Arguments:

auxilaryBusEnabled

PoExtBus enabled status (if 1, PoExtBus is enabled, otherwise disabled)

dataBytes

Array of 10 byte values that are sent to the bus.

Remarks:

No remarks.

14. Interfacing with PoKeys library – C# example

Preinitialization

C#

1. Add a reference to a PoKeysDevice_DLL.dll, located in installation folder
2. Use the class `PoKeysDevice` from the PoKeysDevice_DLL namespace

Visual Basic 6.0

1. Add a reference to a PoKeysDevice_DLL.tlb, located in installation folder
2. Use the class `PoKeysDevice_DLL.PoKeysDevice`

Class initialization

```
PoKeysDevice_DLL.PoKeysDevice cPoKeys = new PoKeysDevice_DLL.PoKeysDevice();
```

Connecting to USB devices – PoKeys55

Enumerating the USB devices (this step must be taken even if we know exact device user ID!)

```
int iNumDevices = cPoKeys.EnumerateDevices();
```

The command returns the number of USB PoKeys devices detected on the system.

Getting device's serial number, user ID, firmware version and pin count:

```
int iSerialNumber = 0;
int iFirmwareVersion = 0;
int iPinNum = 0;
byte iUserID = 0;


for (int n = 0; n < iNumDevices; n++)
{
   cPoKeys.ConnectToDevice(n);
   cPoKeys.GetDeviceID(ref iSerialNumber, ref iFirmwareVersion, ref iPinNum);
   cPoKeys.GetUserID(ref iUserID);
   cPoKeys.DisconnectDevice();

   Console.WriteLine(n + ". device: Serial: " + iSerialNumber + "     Firmware: " +
iFirmwareVersion + "  User ID: " + iUserID);
}
```

Before any data can be read from or written to the device, the command ConnectToDevice must be executed. It's parameter is a device's index and not the userID! (therefore can be changed when multiple devices are connected at a time).

Connecting to Ethernet devices – PoKeys56E

Enumerating the ethernet devices, connecting and retrieving basic data

The example below enumerates PoKeys56E devices, connects to every one of them and displays its user ID and type in the listbox named lstEthDevices.

```
// Enumerate and list ethernet devices
MyDevice.StartEthernetDiscovery();
byte userID = 0;

lstEthDevices.Items.Clear();
lstEthDevices.Items.Add("Searching....");
lstEthDevices.Enabled = false;
lstEthDevices.Refresh();
Application.DoEvents();
Thread.Sleep(2000);

MyDevice.StopEthernetDiscovery();

if (MyDevice.GetNumberOfDetectedNetworkDevices() > 0)
{
   lstEthDevices.Enabled = true;
   lstEthDevices.Items.Clear();
   cmdEthConnect.Enabled = true;
}
else
{
   lstEthDevices.Items.Clear();
   lstEthDevices.Items.Add("No device detected");
   cmdEthConnect.Enabled = false;
}
for (int k = 0; k < MyDevice.GetNumberOfDetectedNetworkDevices(); k++)
{
     // Three possibilites to connect to network device...
    MyDevice.ConnectToNetworkDevice(MyDevice.EthDeviceIP[k]);

    // Also
    //MyDevice.ConnectToNetworkDevice(MyDevice.GetNetworkDeviceAddress(k));

    // or even (with pointers)
    //PoKeysDevice_DLL.sPoKeysNetworkDeviceData netData = new
PoKeysDevice_DLL.sPoKeysNetworkDeviceData();
    //IntPtr ptr =  Marshal.AllocHGlobal(200);
    //Marshal.StructureToPtr(netData, ptr, false);

    //MyDevice.COM_GetNetworkDeviceData(ptr, 0);

    //netData = (PoKeysDevice_DLL.sPoKeysNetworkDeviceData)Marshal.PtrToStructure(ptr,
typeof(PoKeysDevice_DLL.sPoKeysNetworkDeviceData));
    //MyDevice.ConnectToNetworkDevice(netData.IPAddress);

   MyDevice.GetUserID(ref userID);
   lstEthDevices.Items.Add(MyDevice.GetDeviceTypeName() + " (" + userID + ")");

   MyDevice.DisconnectDevice();
}
```

Remarks: the array `MyDevice.EthDeviceIP` holds the IP addresses of all detected devices in the process of automatic discovery.

PoKeys56E has a short connection timeout setting of 3 seconds. After no data is received for this period, the connection with the host is terminated.

Reading pin configuration

```
byte iPinFunction = 0;

cPoKeys.GetPinData(0, ref iPinFunction);
```

In this example 0 (Pin 1) is used for a pin ID. Pin IDs are 0 based.

iPinFunction has the following structure

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Pin invert	reserved	reserved	A output	A input	D output	D input	reserved

where A stands for analog and D for digital.

Reading pin key mapping (only on USB PoKeys devices – PoKeys55)

Let us presume that pin 2 is defined as keyboard digital input with direct key mapping

```
byte iPinKey = 0;
byte iPinModifier = 0;
byte iMappingType = 0;

cPoKeys.GetPinKeyMapping(1, ref iMappingType, ref iPinKey, ref iPinModifier);
```

iPinKey is a key code as described in USB HID standard
iPinModifier is a modifier for a key (Ctrl, Alt, Shift, Win key) and can be used with these masks:

```
const byte CtrlMask = 1;
const byte ShiftMask = 2;
const byte AltMask = 4;
const byte WinMask = 8;
const byte AltGrMask = 64;
```

iMappingType has the following structure

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
reserved	reserved	reserved	reserved	reserved	macro	direct	enable

bit 0 – Enable key mapping: to enable key mapping for a specific pin, this bit must be set to 1
bit 1 – enable direct key mapping: when this bit is set to 1, pin actions are directly reflected as a keyboard key
bit 2 – enable keyboard macro mapping: when this bit is set to 1, special macro sequence is sent on pin activation

There can be only one of the bits 1 or 2 set!

Reading input value

Let us presume that pin 3 is configured as digital input.

```
bool bInputVal = false;
cPoKeys.GetInput(2, ref bInputVal);
Console.WriteLine("Input 3 is " + (bInputVal?"On":"Off"));
```

Joystick axis mapping (only on USB PoKeys devices – PoKeys55)

This is only possible on pins 43-47. If this is used on any other pin, the function will fail or be ignored.

```
cPoKeys.SetJoystickAxisMapping(42, iJoystickAxis);
```

iJoystickAxis can be used as follows:

```
0  None
1  Rx
2  Ry
3  X
4  Y
5  Throttle
```

Block read - digital

It is possible to poll 32 input pins with one request. All 55 pins can be read with two request joined in single command.

```
// Read pins 1 to 32
bool[] values_1_32 = new bool[32];

myDevice.BlockGetInput1(ref values_1_32);

// Read pins 33 to 55
bool[] values_33_55 = new bool[23];

myDevice.BlockGetInput2(ref values_33_55);

// Read all pins (1-55)
bool[] values = new bool[55];

myDevice.BlockGetInputAll55(ref values);
```

Block read - analog

It is possible to poll 4 8-bit or 3 10-bit analog inputs with one command.

```
// 8-bit mode
byte[] channels = { 42, 43, 0, 45 };
byte[] values = new byte[channels.Length];

myDevice.BlockGetAnalogInput8bit(ref channels, ref values);

byte value1 = values[0];
byte value2 = values[1];
byte value3 = values[3];

// 10-bit mode
byte[] channels = { 42, 43, 45};
int[] values = new int[channels.Length];

myDevice.BlockGetAnalogInput10bit(ref channels, ref values);

int value1 = values[0];
int value2 = values[1];
int value3 = values[2];
```

Block write - digital

It is possible to set 32 output pins with one request. All 55 pins can be set with two request joined in single command.

```
// Simple 8-bit binary counter
bool[] states = new bool[32];

for (int n = 0; n < 255; n++)
{
  for (int i = 0; i < 8; i++)
```

```
  {
    if ((n & (1 << i)) > 0) states [i] = false; else states [i] = true;
  }
  MyDevice.BlockSetOutput1(ref states); // Update pins 1 to 32
}
```

Reading encoder RAW values

RAW values from the encoder inputs can be read with following command.

```
byte iEncoderValue = 0;

cPoKeys.GetEncoderValue(1, ref iEncoderValue);
```

iEncoderValue is a value between 0 and 255.

Macro operations (only on USB PoKeys devices – PoKeys55)

Create new macro

This command creates macro in first free position. It returns macro index.

```
byte iMacroID = 0;
byte iMacroLen= 10;

cPoKeys.MacroCreate(iMacroLen, ref iMacroID);
```

Modify macro length

This command modifes macro length.

```
byte iMacroID = 0;
byte iMacroNewLen = 50;

cPoKeys.MacroModifyLength(iMacroID, iMacroNewLen);
```

Delete macro

This command deletes specific macro.

```
byte iMacroID = 0;

cPoKeys.MacroDelete(iMacroID);
```

Save macro configuration to flash

This command saves the current macro configuration to flash.

```
cPoKeys.MacroSaveConfiguration();
```

Change macro name

This command changes the macro name. Name property supports up to 7 characters.

```
byte iMacroID = 0;
cPoKeys.MacroSetName(iMacroID, "Macro1");
```

Set macro key

This command sets one macro key at the position iIndex. This index must be between 0 and iMacroLen - 1.

```
byte iIndex = 5;
byte iKeyCode = 10;
byte iKeyModifier = 0;

cPoKeys.MacroSetKey(iMacroID, iIndex, iKeyCode, iKeyModifier);
```

Get free space for macros

Space for saving macros is limited. To find out how much free space exists, use the following command.

```
int iFreeSpace = 0;

cPoKeys.MacroGetFreeSpace(ref iFreeSpace);
```

Get the list of macros' states

If the macro has the length of 0 it is designated as inactive. This command retrieves the list of states for all the macros. If specific macro is active, bActiveMacros has the value True.

```
bool[] bActiveMacros = new bool[64];

cPoKeys.MacroGetActiveMacros(ref bActiveMacros);
```

Display current time on LCD

```
// Initialize library
PoKeysDevice_DLL.PoKeysDevice dev = new PoKeysDevice_DLL.PoKeysDevice();

// Enumerate devices and connect to first (we have only 1 connected)
dev.EnumerateDevices();
dev.ConnectToDevice(0);

// Set settings for 4x20 LCD
dev.LCDSetSettings(1, 4, 20);
// Initialize LCD
dev.LCDInit();
// Clear LCD
dev.LCDClear();
// Move cursor to home
dev.LCDGotoXY(1, 1);
// Print Hello, world!
dev.LCDPrint("Hello, world!");

// 1s delay
System.Threading.Thread.Sleep(1000);

while (true)
{
  Application.DoEvents();
  System.Threading.Thread.Sleep(100);

  // Move cursor to home
  dev.LCDGotoXY(1, 1);
  // Print current date and time
  dev.LCDPrint(DateTime.Now.ToString());
}
```

Configuring PWM outputs

Configures pin 17 as PWM output with 20 ms period and 1.5 ms (7.5%) duty cycle. If a model RC servo signal input is connected to this pin, servo motor horn should position itself in the middle position.

```
bool[] channel = new bool[6];
uint[] duty = new uint[6];
```

```
// PWM base clock is 12 MHz (or 25 MHz on PoKeys56 devices), so 1 ms takes 12000 (25000)
cycles
float ms = MyDevice.GetPWMFrequency() / 1000;
uint period = (uint)(ms * 20);   // 20 ms period

channel[5] = true; // Pin 17 = channel 5 (Pin 18 = channel 4, ...)
duty[5] = (uint)(ms * 1.5); // Set duty cycle to 1.5 ms (7.5 %)

MyDevice.SetPWMOutputs(ref channel, ref period, ref duty);
```

Configuring matrix keyboard

Configures a 4x4 matrix keyboard, connected to first eight pins of PoKeys device.

```
byte[] rows = new byte[16]; // This must always be the size of 16
byte[] cols = new byte[8];  // This must always be the size of 8

bool[] macros = new bool[128];      // This must always be the size of 128
byte[] keys = new byte[128];         // This must always be the size of 128
byte[] modifiers = new byte[128];    // This must always be the size of 128

for (int i=0; i<4; i++)
{
    cols[i] = (byte)i;
    rows[i] = (byte)(i+4);
}

byte configuration = 1;
byte numrows = 4;
byte numcols = 4;

// Set column pins as digital inputs
MyDevice.SetPinData(0, 1 << 1);
MyDevice.SetPinData(1, 1 << 1);
MyDevice.SetPinData(2, 1 << 1);
MyDevice.SetPinData(3, 1 << 1);

// Set row pins as digital outputs
MyDevice.SetPinData(4, 1 << 2);
MyDevice.SetPinData(5, 1 << 2);
MyDevice.SetPinData(6, 1 << 2);
MyDevice.SetPinData(7, 1 << 2);

MyDevice.SetMatrixKeyboardConfiguration(ref configuration, ref numcols, ref numrows, ref
rows, ref cols, ref macros, ref keys, ref modifiers);
```

Reading matrix keyboard status

Reads all matrix keyboard keys statuses.

```
bool[] KeyStates = new bool[128];

MyDevice.GetMatrixKeyboardKeyStatus(ref KeyStates);

string status = "";
for (int row = 0; row < 4; row++)
{
    for (int col = 0; col < 4; col++)
    {
        status += KeyStates[row * 8 + col] ? "1 " : "0 ";
    }
    status += "\n";
}

MessageBox.Show("Key statuses:\n" + status);
```

Write data to PoExtBus device

Sends the byte to the first PoExtBus device. As the devices can be daisy-chained, the first device connected to PoKeys device has the index 9. If any device is daisy-chained to this device, it would have the index 8, etc.

```
byte[] dataOut = new byte[10];

// Set some value
dataOut[9] = 0xAA;

MyDevice.AuxilaryBusSetData(1, dataOut);
```

Write data to matrix LED display

This example activates the matrix LED display 1 as 8x8 display, clears it and draws a + sign on it.

```
// Enable LED1
MyDevice.MatrixLEDSetSettings(true, 8, 8, false, 8, 8);

// Invert the whole matrix
MyDevice.MatrixLED1ClearAll(true);

// Draw a dark + sign
MyDevice.MatrixLED1SetPixel(2, 1, false);
MyDevice.MatrixLED1SetPixel(1, 1, false);
MyDevice.MatrixLED1SetPixel(0, 1, false);
MyDevice.MatrixLED1SetPixel(1, 0, false);
MyDevice.MatrixLED1SetPixel(1, 2, false);
```

Read tick counter

Every PoKeys device has a tick counter, a counter that internally counts milliseconds.

```
uint ticks = 0;

MyDevice.GetTickCounter(ref ticks);

MessageBox.Show("Tick counter: " + ticks);
```

Read temperature from the LM75 sensor, connected to I²C bus

```
byte stat = 0;
byte[] data = new byte[32];

// Turn off PoExtBus
MyDevice.AuxilaryBusSetData(0, data);

// Activate I2C
MyDevice.I2CSetStatus(true);

// Write 0 to the device at the address 0x90 (LM75 command: set address)
data[0] = 0;
MyDevice.I2CStartWrite(0x90, 1, data);
MyDevice.I2CGetWriteStatus(ref stat);

if (stat != 1)
{
    MessageBox.Show("Error writing");
    return;
}

// Read temperature - read two bytes from the device at the address of 0x90
MyDevice.I2CStartRead(0x90, 2);
// This operation was quite fast, so no wait is needed...
MyDevice.I2CGetReadStatus(ref stat, ref data);

if (stat == 1)
{
    MessageBox.Show("Temperature: " + data[0] + "°C");
}
else
```

```
{
    MessageBox.Show("Error reading");
}

// Deactivate I2C
MyDevice.I2CSetStatus(false);
```

Read temperature from DS18B20 sensor, connected to 1-wire bus

```
byte stat = 0;
byte[] data = new byte[16];

MyDevice.prot1WireSetStatus(true);

data[0] = 0xCC;
data[1] = 0x44;
MyDevice.prot1WireStartWriteAndRead(2, 0, data);

Thread.Sleep(1000);

data[0] = 0xCC;
data[1] = 0xBE;
MyDevice.prot1WireStartWriteAndRead(2, 9, data);

Thread.Sleep(10);

byte len = 0;
MyDevice.prot1WireGetReadStatus(ref stat, ref data, ref len);

if (stat == 1)
{
    MessageBox.Show("Temperature: " + (((data[1] << 8) + data[0]) >> 4) + "°C");
}
else
{
    MessageBox.Show("Error reading");
}
```

15. Major changes from 1.x to 1.7:

To move the PoKeys55 device to a new level some major changes to interface were imminent.

Pi n function 1 was removed (this was directly key mapped pin function). Instead, key mapping functionality was added to any digital input pin. Key mapping type (none/direct/macro) can be set via changed Key mapping command as shown in above example.

Pin 13 not functioning appropriately

On PoKeys55 devices with serial numbers greater than 10133 and lower than 11500 there is a flawed connection for pin 13. Please do not use this pin.

Note: on PoKeys55 boards with serial number above 11500, this problem is removed.

Putting pin 4 low on startup disables PoKeys55 device from booting.

Avoid connecting switches that can be closed on startup to this pin.

Note: on PoKeys55 boards with serial number above 11500, this problem is removed.

16. Errata information

This section describes special limitations of the device.

16.1.1. Pins 5 and 6 cannot be separately set as outputs/inputs

Pins 5 and 6 must be both set to either input or output

Affected: PoKeys56E

17. Grant of license

The material contained in this release is licensed, not sold. PoLabs grants a license to the person who installs this software, subject to the conditions listed below.

1. Access

The licensee agrees to allow access to this software only to persons who have been informed of and agree to abide by these conditions.

2. Usage

The software in this release is for use only with PoLabs products or with data collected using PoLabs products.

3. Copyright

PoLabs claims the copyright of, and retains the rights to, all material (software, documents etc) contained in this release. You may copy and distribute the entire release in its original state, but must not copy individual items within the release other than for backup purposes.

4. Liability

PoLabs and its agents shall not be liable for any loss or damage, howsoever caused, related to the use of PoLabs equipment or software, unless excluded by statute.

5. Fitness for purpose

No two applications are the same, so PoLabs cannot guarantee that its equipment or software is suitable for a given application. It is therefore the user's responsibility to ensure that the product is suitable for the user's application.

6. Mission Critical applications

Because the software runs on a computer that may be running other software products, and may be subject to interference from these other products, this license specifically excludes usage in 'mission critical' applications, for example life support systems.

7. Viruses

This software was continuously monitored for viruses during production, however the user is responsible for virus checking the software once it is installed.

8. Support

No software is ever error-free, but if you are unsatisfied with the performance of this software, please contact our technical support staff, who will try to fix the problem within a reasonable time.

9. Upgrades

We provide upgrades, free of charge, from our web site at www.poscope.com. We reserve the right to charge for updates or replacements sent out on physical media.

10. Trademarks

Windows is a registered trademark of Microsoft Corporation. PoKeys, PoKeys55, PoKeys56E, PoScope, PoLabs and others are internationally registered trademarks.

support: www.poscope.com